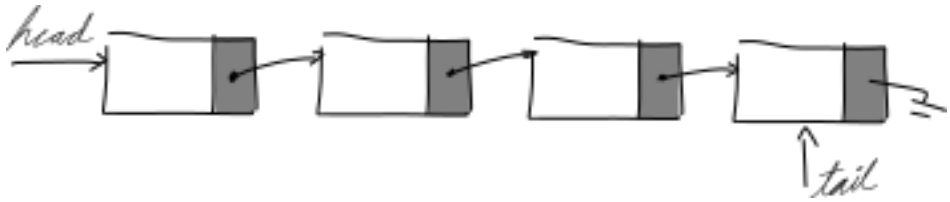
Facultad de Estudios Superiores

Acatlán

Matemáticas Aplicadas y Computación

Estructura de datos

SYLLABUS 2026 – 1



Jesús Daniel Cordero Vázquez

900949@pcpuma.acatlan.unam.mx

1 Introducción

La materia de Estructura de Datos es crucial en la ingeniería de software y la ciencia de la computación porque enseña a los estudiantes cómo organizar y gestionar datos de manera eficiente. Su objetivo principal es resolver problemas de manera óptima, lo que impacta directamente en la velocidad y el rendimiento de los programas. Sin un conocimiento sólido de esta materia, un desarrollador solo puede escribir código que funciona, pero no código que sea eficiente y escalable para aplicaciones del mundo real.

La importancia de las estructuras de datos se debe al amplio margen de aplicaciones en el mundo tecnológico, como lo son

- **Bases de datos**: Las bases de datos utilizan estructuras como árboles B y tablas hash para indexar y buscar datos de forma rápida.
- **Sistemas operativos**: El manejo de procesos y recursos se gestiona con estructuras como listas enlazadas y colas (queue).
- **Inteligencia artificial y Machine Learning**: Los algoritmos de aprendizaje automático utilizan estructuras de datos complejas para almacenar y procesar grandes conjuntos de datos.
- **Gráficos por computadora:** Las estructuras como los árboles se utilizan para la detección de colisiones y la representación de espacios 3D.

Por ello, la materia de estructura de datos tiene un gran impacto más allá del aula, volviéndose en una herramienta fundamental para el egresado de la licenciatura en Matemáticas Aplicadas y Computación.

1.1 Ubicación en el mapa curricular y requisitos previos

La asignatura de Estructura de Datos es de tercer semestre y de carácter obligatorio para la licenciatura en Matemáticas Aplicadas y Computación para el tercer semestre de la carrera.

1.2 Objetivo general

El alumno aplicará las estructuras de datos, así como las técnicas de recuperación y ordenamiento de datos, en la implementación de algoritmos computacionales.

1.3 Objetivos específicos

- **Comprender la relación entre datos y algoritmos**: Estudiar cómo la elección de una estructura de datos impacta directamente en la eficiencia de un algoritmo. El estudiante aprenderá que no existe una única estructura de datos ideal para todos

los problemas, sino que la elección depende de las operaciones que se realicen con mayor frecuencia (inserción, eliminación, búsqueda, etc).

- **Analizar la complejidad algorítmica**: El estudiante aprenderá a usar la notación de la gran O (Big O) para analizar el rendimiento y el uso de memoria de los algoritmos. Este objetivo les permite predecir el comportamiento de un programa a medida que el conjunto de datos crece, y así seleccionar la solución más eficiente para un problema.
- **Dominar las estructuras de datos fundamentales**: Adquirir un conocimiento profundo de estructuras como listas, pilas, colas, árboles (binarios, de búsqueda, AVL, B-trees) y tablas hash. El estudiante aprenderá no solo a implementar estas estructuras, sino también a entender sus propiedades, ventajas y desventajas.
- **Desarrollar habilidades de resolución de problemas**: Fomentar el pensamiento lógico y analítico para abordar problemas complejos. El estudio de las estructuras de datos entrena a los estudiantes para modelar problemas del mundo real y transformarlos en representaciones de datos, eligiendo la estructura más adecuada para lograr una solución óptima y eficiente.
- **Preparar para el desarrollo profesional**: El estudiante contará con el conocimiento necesario para trabajar en áreas de la informática que requieren un manejo eficiente de grandes volúmenes de datos, como la ingeniería de software, la inteligencia artificial o el análisis de datos. Las estructuras de datos son un tema recurrente en las entrevistas técnicas de las principales empresas tecnológicas.

1.4 Horario y salón de clases

Grupo 1301

Semestre agosto – diciembre 2025

Lunes – 07:00 a 09:00 horas | Laboratorio A-723

Miércoles y viernes - 07:00 - 09:00 horas | Salón A-405

2 Temario

Tema y subtemas	Objetivos específicos
1. Introducción a los algoritmos de estructuras de datos 1.1. Abstracción de datos 1.2. Clasificación de las estructuras de datos 1.3. Algoritmos y estructuras de datos 1.4. Técnicas de diseño de algoritmos 1.5. Recursividad	El objetivo principal de esta sección es que los estudiantes comprendan los conceptos fundamentales que rigen el diseño y la eficiencia de los algoritmos y estructuras de datos. Esto les permitirá sentar las bases para los temas más avanzados del curso. • El estudiante diferenciará un tipo de dato abstracto (TDA) de una estructura de datos lineal o no lineal, estática y dinámica, así como sus ventajas y desventajas, enfocándose en cómo el TDA define el comportamiento de los datos sin detallar su implementación interna. • Los estudiantes analicen la eficiencia de los algoritmos utilizando la notación de Big O (O) para predecir el rendimiento de un programa en función de su tamaño de entrada. • Los estudiantes comprendan y apliquen la recursividad, distinguiendo entre una solución recursiva y una iterativa. Deben ser capaces de identificar los casos base y recursivos para resolver problemas correctamente.
2. Almacenamiento estático, dinámico y estructuras elementales 2.1 Almacenamiento estático y dinámico 2.2 Arreglos 2.2.1 Arreglos de diferentes tipos de datos 2.2.2 Arreglos de punteros 2.2.3 Arreglos como parámetros en funciones 2.2.4 Arreglos unidimensionales, bidimensionales y multidimensionales 2.2.5 Arreglos dinámicos 2.2.6 Arreglos triangulares: inferior y superior 2.3 Registros 2.3.1 Arreglos de registros (estáticos y dinámicos) 2.4 Enumeraciones y colecciones	El objetivo principal de esta sección es que los estudiantes comprendan los conceptos de almacenamiento de memoria y estructuras de datos elementales para manipular información de forma eficiente. Esto les permitirá sentar las bases para estructuras más complejas. • Los estudiantes diferenciarán entre la asignación de memoria estática (en tiempo de compilación) y dinámica (en tiempo de ejecución). Deben comprender sus ventajas, desventajas y saber cómo aplicar cada tipo de almacenamiento en un programa. • Los estudiantes dominarán el uso de arreglos en todas sus formas. Siendo capaces de: ◦ Manejar arreglos de diferentes tipos de datos, incluyendo punteros. ◦ Entender cómo se pasan los arreglos como parámetros a funciones. ◦ Implementar arreglos de una, dos y múltiples dimensiones, así como arreglos dinámicos para manejar conjuntos de datos de tamaño variable. ◦ Comprender y trabajar con arreglos triangulares (inferior y superior) para

	optimizar el almacenamiento de matrices escasas. • Los estudiantes aprenderán a utilizar registros (o structs en C) para agrupar datos de distintos tipos bajo un solo nombre. Esto les permitirá modelar objetos del mundo real y gestionar colecciones de datos complejos. • Los estudiantes comprenderán el propósito de las enumeraciones para crear conjuntos de constantes con nombres. Además, deben familiarizarse con el concepto general de colecciones como contenedores de datos, preparando el terreno para las estructuras lineales que se verán más adelante en el temario.
3. Estructuras de datos lineales y no lineales 3.1 Pilas 3.1.1 Definición y representación 3.1.2 Operaciones de inserción y extracción 3.1.3 Implementación dinámica con diferentes tipos de datos 3.1.4 Implementación con arreglos de diferentes tipos de datos 3.1.5 Notación infija, prefija y postfija. 3.2 Cola 3.2.1 Definición y representación 3.2.2 Operaciones de inserción y extracción 3.2.3 Implementación dinámica con diferentes tipos de datos 3.2.4 Implementación con arreglos de diferentes tipos de datos 3.2.5 Cola circular, doblemente ligada y de prioridad 3.3Listas 3.3.1 Definición y representación 3.3.2 Operaciones de inserción y extracción 3.3.3 Implementación dinámica con diferentes tipos de datos 3.3.4 Implementación con arreglos de diferentes tipos de datos	El objetivo de esta sección es que los estudiantes dominen las principales estructuras de datos, pilas, colas, listas y árboles, comprendiendo su lógica, implementación y la mejor manera de aplicarlas para resolver problemas de manera eficiente. • Los estudiantes comprenderán el funcionamiento de las pilas (LIFO, Last-In, First-Out) y las colas (FIFO, First-In, First-Out). Sincod capaces de: • Identificar las operaciones básicas (push, pop en pilas; enqueue, dequeue en colas) y aplicarlas en la programación. • Implementar ambas estructuras usando asignación de memoria estática (con arreglos) y dinámica (con listas enlazadas), y entender las ventajas de cada enfoque. • Reconocer la importancia de la notación de expresiones matemáticas (infija, prefija y postfija) y cómo las pilas son fundamentales para su evaluación. • Conocer las variantes de las colas, como la circular y la doble, para optimizar su uso en diferentes escenarios. • Los estudiantes dominarán el uso de listas enlazadas. Siendo capaces de: • Comprender la diferencia entre una lista y un arreglo. • Implementar operaciones como inserción, extracción y búsqueda. • Dominar los diferentes tipos de listas enlazadas: simplemente enlazadas, doblemente enlazadas y circulares, así

3.3.5 Ligadas, doblemente ligadas, circulares y ordenadas 3.4 Árboles 3.4.1 Definición y representación 3.4.2 Clasificación, operaciones de inserción, extracción, búsqueda y recorrido 3.4.2.1 Árboles Binarios 3.4.2.2 Árboles AVL 3.4.2.3 Árboles B 3.4.3 Implementación de árboles con diferentes tipos de datos 3.4.4 Bosques	como las listas ordenadas, y conocer sus respectivos casos de uso. • Los estudiantes entenderán la lógica y el funcionamiento de las estructuras de datos no lineales, específicamente los árboles. Siendo capaces de: • Definir un árbol y sus componentes principales (nodo, raíz, rama, hoja). • Implementar las operaciones básicas como la inserción, extracción, búsqueda y los recorridos (preorden, inorden, postorden). • Distinguir entre diferentes tipos de árboles como los árboles binarios, árboles AVL (para mantener el balance) y árboles B (utilizados en bases de datos). • Comprender el concepto de bosque como una colección de árboles.
4. Técnicas de ordenamiento y búsqueda 4.1 Métodos de ordenación 4.1.1 Selección 4.1.2 Inserción (Shell) 4.1.3 Intercambio 4.1.4 Mezcla 4.1.5 Rápida (quick sort) 4.1.6 Polifase 4.1.7 Cascada 4.1.8 Oscilante 4.2 Técnicas de búsqueda 4.2.1 Comparación de llaves (lineal y binaria) 4.2.2 Transformación de llaves (funciones de Hash y colisiones)	El objetivo de esta unidad es que los estudiantes comprendan, apliquen y analicen las principales técnicas para organizar y encontrar datos. Se espera que puedan seleccionar el método más eficiente según el contexto y el tipo de datos. • Los estudiantes dominarán los diferentes algoritmos de ordenamiento, comprendiendo su lógica y sus respectivas complejidades. Siendo capaces de: • Identificar y aplicar métodos de ordenamiento básicos como selección, inserción e intercambio (burbuja). • Implementar y evaluar algoritmos de ordenamiento más avanzados y eficientes, como Shell sort (variante de inserción), mergesort (mezcla) y quicksort (rápida). • Conocer los algoritmos de ordenamiento externo, como polifase, cascada y oscilante, y entender cómo se usan para ordenar grandes volúmenes de datos que no caben en la memoria principal. • Los estudiantes comprenderán cómo encontrar datos de manera eficiente. Siendo capaces de: • Comparar las técnicas de búsqueda por llaves: búsqueda lineal y búsqueda binaria. Deben saber cuándo usar cada una y entender por qué la búsqueda

	binaria es significativamente más rápida en arreglos ordenados. • Comprender el concepto de funciones de hash para la búsqueda directa, y cómo se implementan. • Analizar el problema de las colisiones y aplicar métodos para resolverlas, como el encadenamiento o el sondeo lineal.

3 Estrategia instruccional

El trabajo presencial se apoyará en la plataforma de Github ubicado en https://github.com/dda-ex/ED-2026-1/tree/main con el fin de brindar un espacio de comunicación, facilitar la distribución de materiales de aprendizaje, así como la realización de prácticas guiadas o entregables.

Es importante mencionar que dada la naturaleza de la materia se estarán llevando dos estrategias instruccionales durante el desarrollo de la materia, existiendo actividades durante la clase y el laboratorio mutuamente complementarias.

El objetivo instruccional durante la clase es introducir los conceptos fundamentales de manera clara y concisa.

- **Fundamentos**: Comenzar con una revisión de los tipos de datos abstractos (TDA) y su diferencia con las estructuras de datos. Esto ayuda al estudiante a comprender la lógica detrás de la implementación.
- **Análisis de la Complejidad**: Introducir la notación de Big O (O) para que el estudiante pueda evaluar y comparar la eficiencia de los algoritmos y estructuras. Es crucial la comprensión de por qué una solución es mejor que otra en términos de tiempo y espacio.
- **Ejemplos reales**: Usar analogías y ejemplos del mundo real para explicar las estructuras. Por ejemplo, una cola se puede comparar con la fila de un supermercado y una pila con un analizador de operaciones básicas.

El objetivo instruccional durante la clase en el laboratorio práctico es aplicar los conceptos a un lenguaje de programación para la resolución de problemas y consolidar el conocimiento.

- **Implementación guiada**: El estudiante debe implementar desde cero las estructuras de datos básicas, como listas enlazadas, pilas, colas y árboles. Esto les ayuda a entender cómo funcionan internamente.

- **Proyectos por unidad**: Dividir el curso en módulos de estructuras de datos (lineales, no lineales, etc.), y asignar proyectos específicos para cada uno. Por ejemplo, un proyecto podría ser la implementación de un sistema de gestión de memoria y recursos, o una calculadora de notación polaca.
- **Resolución de problemas**: Presentar problemas de programación que requieran la elección y el uso de la estructura de datos más adecuada. Estos problemas deben ir de lo simple a lo complejo.

4 Evaluación

La evaluación del curso será la siguiente:

Equivalencia	Apartado
45%	Exámenes teóricos
30%	Tareas y programas
25%	Proyecto final

El examen teórico se compone de 3 exámenes en el cual se contemplan lo siguiente.

Equivalencia	Examen	Temas	Comentarios	Fecha
15%	Primer examen	Unidad 1. Introducción a los algoritmos de estructuras de datos Unidad 2. Almacenamiento estático, dinámico y estructuras elementales	Mínimo aprobatorio 7	Viernes 2 de septiembre de 2025
15%	Segundo examen	Unidad 3. Estructuras de datos lineales y no lineales	Mínimo aprobatorio 7	Viernes 24 de octubre de 2025
15%	Tercer examen	Unidad 4. Técnicas de ordenamiento y búsqueda	Mínimo aprobatorio 7	Viernes 21 de noviembre de 2025

Es importante mencionar que las fechas son propuestas y podrían cambiar de acuerdo con el avance del grupo y desarrollo de actividades, se informará al alumnado con anticipación la fecha de aplicación.

Notas:

Tarea_copiada = anulacion_para_los_implicados - La vida es muy corta para preocuparse por atribuir responsabilidades.

Uso de herramientas de AI para creación del tareas y programas está totalmente prohibido por lo que la actividad será anulada.

Todos los programas, así como el proyecto final, deben hacer hacer uso de los archivos creados en clase.

5 Referencias

5.1 Libros de texto

- La bibliografía del curso comprende los siguientes libros: Understanding and Using C Pointers - Richard Reese http://www.sauleh.ir/fc98/static_files/materials/Richard%20Reese-Understanding%20and%20Using%20C%20Pointers-O%27Reilly%20Media%20(2013).pdf
- Introduction to Algorithms, Cormen
- Cairó, Osvaldo & Guardati, Silvia. (2006). Estructuras de datos (3 ed). México: McGraw-Hill

5.2 Referencias básicas

- Bowman, C. (1999). Algoritmos y estructuras de datos. México: Oxford.
- Cairo y Guardati. (1999). Estructuras de datos. México: McGraw Hill.
- Cairó, Osvaldo & Guardati, Silvia. (2006). Estructuras de datos (3 ed). México: McGraw-Hill.
- Guardati, Silvia. (2007). Estructura de Datos Orientada a Objetos. México: Pearson.
- Knuth, D. (2002). El arte de programar ordenadores, Algoritmos fundamentales (Vol. 1). Barcelona: Reverte.
- Koffman, Elliot B. & Wolfgang, Paul A.T. (2008). Estructura de datos con C++. Objetos, abstracciones y diseño. México: McGraw-Hill.
- Kruse, et al. (1999). Data structures and program design in C++. E. U. A.: Pretince Hall.
- Lipschutz, S. (1987). Estructura de datos. México: McGraw Hill.
- Sedgewick, R. (1992). Algorithms in C++. E.U.A: Addison Wesley.
- Tenenbaum, et al. (1993). Estructura de datos en C. E.U.A.: Prentice Hall.
- Weiss, M. (1992). Data structures and algorithm analysis. E.U.A.: Addison Wesley.

- Weiss, Mark Allen (2006). Data Structures & Problem Solving Using Java (3ed.). E.U.A. Addison Wesley.
- Wirth, N. (1992) Algoritmos+ estructuras de datos = programas. España: Castillo.

5.3 Referencias complementarias

- Aho, et al. (1988). Estructura de datos y algoritmos. E.U.A.: Addison Wesley.
- Heileman, G. (1997). Estructura de datos, algoritmos y programación orientada a objetos. México: McGraw Hill.
- Knuth, D. (1980). Algoritmos fundamentales. México: Reverte.
- Wirth, N. (1987). Algoritmos y estructura de datos. México: Prentice Hall.

6 Contacto y comunicación

Jesús Daniel Cordero Vázquez, 900949@pcpuma.acatlan.unam.mx

También es posible establecer comunicación mediante el github en el apartado de *issues*.