

UNIVERSIDAD NACIONAL DE INGENIERÍA

SÍLABO

INFORMACIÓN GENERAL

ASIGNATURA	:ESTRUCTURAS DE DATOS
DOCENTE	:GERSON GARRIDO
HORAS POR SEMANA	:06 (TEORÍA: 02, LABORATORIO: 04)

SUMILLA

El curso es de naturaleza teórico-práctica y enseña a organizar los datos para que un programa siga siendo rápido cuando los datos crecen. El estudiante construye las estructuras que las bibliotecas traen hechas, predice su costo y lo mide. Se organiza en cuatro partes: **de C a C++ y costos** (clases, ownership, templates, análisis asintótico y medición); **secuencias y ordenamiento** (arreglos dinámicos, listas, pilas, colas y ordenamiento); **diccionarios** (tablas hash, árboles de búsqueda, AVL y rojinegros); y **prioridades y grafos** (heaps, grafos, union-find y caminos mínimos). El laboratorio se desarrolla en C++, en un entorno común.

COMPETENCIAS

- Escribe clases y templates en C++ que administran su memoria con RAII, sin fugas ni accesos inválidos.
- Distingue un tipo abstracto de datos de sus implementaciones, y elige la adecuada según las operaciones que el problema necesita.
- Predice el costo de un programa con análisis asintótico y costo amortizado, y contrasta la predicción con mediciones.
- Implementa y usa arreglos dinámicos, listas enlazadas, pilas, colas y deques.
- Implementa y compara tablas hash con encadenamiento y con direccionamiento abierto.
- Implementa árboles binarios de búsqueda y los mantiene balanceados con AVL y con árboles rojinegros.
- Implementa heaps y colas de prioridad, y representa y recorre grafos.
- Declara los invariantes de una estructura, los comprueba en el código y la prueba contra una implementación de referencia.
- Explica cómo las bibliotecas y los lenguajes actuales aplican estas ideas.

UNIDADES DE APRENDIZAJE

Unidad	Contenido	Semanas
I	De C a C++ y costos	1–3
II	Secuencias y ordenamiento	4–6
III	Diccionarios	7–12
IV	Prioridades y grafos	13–16

PROGRAMACIÓN SEMANAL

Cada semana de clases comprende una sesión de teoría de 2 horas y dos sesiones de laboratorio de 2 horas cada una. Cada capítulo de los apuntes empieza con una ruta de lectura: qué leer antes de clase (15 a 20 minutos), qué se explica en clase y qué repasar después. Los temas marcados con (*) pueden pasar a lectura autónoma si se pierden clases por feriados.

Sem.	Teoría	Laboratorio
1	Presentación. De C a C++: referencias, clases, `std::string`, `std::vector` y pruebas	Portar un módulo de C a C++
2	Ownership y RAII: destructor, copia y movimiento. Templates. Tipo abstracto e implementación, con la pila como ejemplo	Una pila genérica sin fugas
3	Análisis asintótico. Medir tiempos	Predecir el crecimiento y medirlo
4	Arreglos dinámicos y costo amortizado. Colas sobre un buffer circular	Un `Vector` propio. *Práctica 1*
5	Listas enlazadas e iteradores. Deques	Un deque con dos representaciones (buffer circular y lista enlazada), medidas
6	Ordenamiento: mergesort y quicksort. Búsqueda binaria	Autocompletado
7	Tablas hash: funciones hash y encadenamiento	La tabla de variables, ahora como tabla hash. *Práctica 2*
8	**Examen parcial**	
9	Direccionamiento abierto: linear probing y tombstones. Swiss Tables y el `dict` de CPython	Corrector ortográfico: comparar diseños
10	Árboles binarios de búsqueda: recorridos y mapas ordenados	Un mapa ordenado, probado contra `std::map`
11	Árboles AVL: rotaciones y altura	Balancear el mapa con AVL. *Práctica 3*
12	Árboles rojinegros: de los árboles 2-3-4 a la inserción. B-trees como idea. Eliminación (*)	Inserción rojinegra. AVL, rojinegro y `std::map`, medidos
13	Heaps y colas de prioridad. Heapsort	Compresión de Huffman
14	Grafos: representación, BFS y DFS. Union-find	Generar y resolver laberintos. *Práctica 4*
15	Dijkstra con un heap. Tries (*). Filtros de Bloom (*). Repaso	Rutas sobre un mapa real
16	**Examen final**	

METODOLOGÍA

- **Teoría:** clases expositivas con ejercicios resueltos en clase y programas seguidos paso a paso. Antes de cada clase, el estudiante lee las secciones de los apuntes que indica la ruta de lectura del capítulo.

- **Laboratorio:** dos sesiones semanales en un entorno común para todos los estudiantes, donde construyen la estructura de la semana, la prueban y la miden en un programa real.
- **Prácticas:** cuatro prácticas individuales, en las semanas 4, 7, 11 y 14, con pruebas automáticas. Esa semana, una de las dos sesiones de laboratorio es para la práctica y la otra sigue con el laboratorio de la guía.
- Los términos técnicos se presentan en español cuando así se dicen (variable, arreglo, función) y en inglés cuando la traducción suena forzada (bit, commit, router).

NORMAS DEL CURSO

- **Trabajo:** individual. Se puede discutir ideas, pero cada estudiante escribe su propio código y debe poder explicarlo. Copiar soluciones de otras personas o de internet cuenta como plagio.

USO DE HERRAMIENTAS E IA GENERATIVA

Las prácticas y los exámenes miden lo que cada estudiante sabe hacer solo, así que ahí no se usa IA generativa. En los laboratorios y para estudiar sí se puede, con estas condiciones:

Herramienta	**Laboratorios y estudio**	**Prácticas**
Mensajes del compilador, `gdb`, valgrind y sanitizers (ASan, UBSan)	Sí	Sí
Documentación oficial (cppreference, manuales de las herramientas) y búsqueda web	Sí	Sí, para consultar el lenguaje y la biblioteca, no para buscar soluciones
Autocompletado del editor que completa nombres y firmas, sin escribir soluciones enteras	Sí	Sí
Corrector ortográfico, traductores y herramientas de accesibilidad	Sí	Sí
Formateadores de código y linters	Sí	Sí
IA generativa para pedir explicaciones, sin que genere código	Sí	No
IA generativa que escribe código (chats, Copilot y similares)	Sí, declarándolo	No

- **Cómo se declara:** quien use IA generativa en un laboratorio lo dice en una línea al inicio del archivo o del informe que entrega: qué herramienta usó y para qué. Para estudiar no hace falta declararlo.
- **Revisión oral:** cualquier práctica o laboratorio puede incluir una revisión oral breve, en la que cada estudiante explica su código. Quien no pueda explicar lo que entregó no demuestra lo que la tarea pide, aunque el programa funcione.

BIBLIOGRAFÍA

- Apuntes del curso, un capítulo por semana, en el sitio del curso. No se requiere comprar ningún libro.
- Morin, P. *Open Data Structures*. Edición en C++ (beta) o en pseudocódigo. Licencia CC BY. `https://opendatastructures.org/`
- Cormen, T., Leiserson, C., Rivest, R. y Stein, C. *Introduction to Algorithms*, 4.ª edición. MIT Press, 2022. Texto de referencia.

- Sitio del curso: guías semanales, apuntes, diapositivas, código y enunciados.