



RIDAA
Repositorio Institucional
Digital de Acceso Abierto de la
Universidad Nacional de Quilmes



Universidad
Nacional
de Quilmes

Martínez López, Pablo

Estructuras de datos



Esta obra está bajo una Licencia Creative Commons Argentina.
Atribución - No Comercial - Compartir Igual 2.5
<https://creativecommons.org/licenses/by-nc-sa/2.5/ar/>

Documento descargado de RIDAA-UNQ Repositorio Institucional Digital de Acceso Abierto de la Universidad Nacional de Quilmes de la Universidad Nacional de Quilmes

Cita recomendada:

Garcino Ruiz, F., Martínez López, P., Castro, A., Espíndola, C. y Confalonieri, G. (2026). *Estructuras de datos. (Programa)*. Bernal, Argentina: Universidad Nacional de Quilmes. Disponible en RIDAA-UNQ Repositorio Institucional Digital de Acceso Abierto de la Universidad Nacional de Quilmes
<http://ridaa.unq.edu.ar/handle/20.500.11807/6505>

Puede encontrar éste y otros documentos en: <https://ridaa.unq.edu.ar>

PROGRAMA ANALÍTICO DE LA ASIGNATURA

Estructuras de Datos

Modalidad Regular

Departamento de Ciencia y Tecnología

Carreras: Lic. en Bioinformática

Asignatura: Estructuras de Datos

Asignaturas correlativas: Introducción a la Programación

Núcleo al que pertenece: Básico

Docentes: Pablo E. Martínez López, Alejandro Castro, Cristian Espíndola, Gisela Confalonieri, Franco Garcino Ruiz.

Carga horaria total: 144 hs

Año lectivo: 2025 y 2026

Objetivos:

Que quienes cursen la materia:

- Comprenda la noción de dato y de estructuras de datos, y su importancia e interrelación estrecha con la estructura algorítmica de un programa.
- Entienda la diferencia entre acceso aleatorio y acceso secuencial.
- Conozca la idea de interfaz de una estructura de datos, y sea capaz de utilizar productivamente para la solución de problemas.
- Conozca la interfaz de distintas estructuras de datos básicas (pilas, colas, listas, árboles, hashing, etc.) y las utilice adecuadamente
- Comprenda y utilice la noción de estructura contenedora, y la capacidad de realizar combinaciones complejas utilizándolas (i.e. pila de lista de registros, etc.)
- Se familiarice con las nociones de ámbito y de pasaje de parámetros por

valor o referencia.

- Maneje alguno de los principios básicos de diseño de interfases de una estructura de datos (separación en constructores e inspectores de una interfase, ecuaciones entre combinaciones de constructores, etc.), y pueda reconocerlos en situaciones prácticas junto con su utilidad.
- Comprenda el concepto de asignación dinámica de memoria, y pueda hacer programas que hagan un manejo dinámico explícito de memoria en forma adecuada.
- Entienda la noción de implementación de una estructura de datos, y de su eficiencia, y sea capaz de implementar las interfases vistas anteriormente con distintas alternativas variadas en eficiencia.
- Se familiarice con las tareas de compilar y vincular programas para lograr un ejecutable.
- Pueda resolver problemas mediante programas recursivos, y entienda la diferencia entre una resolución recursiva y otra iterativa.

Contenidos mínimos:

- Recursión sobre listas y árboles. Programas recursivos.
- Tipos algebraicos: Maybe, Either, enumerativos, listas, árboles binarios.
- Estructuras contenedoras: pilas, colas, diccionarios, heaps, árboles balanceados.
- Nociones de representación e invariante de representación y su utilidad en el diseño e implementación de estructuras de datos.
- Uso imperativo de estructuras de datos. Iteración en listas y árboles.
- Modelo de memoria imperativo: stack/heap, asignación de memoria. Punteros. Variables por referencia.
- Listas encadenadas y sus variantes. árboles implementados con punteros. Binary heaps implementadas con arrays.
- Hashing. Análisis de eficiencia e implementación.
- Algoritmos de ordenamiento. Clasificación e implementación.

Programa analítico:

Unidad 1: Introducción a Estructuras de Datos

Nociones básicas de estructuras de datos. Tipos algebraicos. Pattern Matching. Ejemplos de tipos algebraicos: Maybe, Either, enumerativos. Recursión. Listas, y árboles.

Unidad 2: Estructuras de datos en alto nivel (puro)

Pilas, Colas, Conjuntos. Nociones de representación e invariante de representación/precondición. Maps. Implementaciones diversas: Listas de pares clave- valor, BSTs, AVLs. Heaps. Binary heaps.

Unidad 3: Introducción a imperativo

Modelo de memoria imperativa, lenguaje C. Memoria Stack y memoria Heap. Alocación de memoria. Punteros. Variables por referencia. Arrays.

Unidad 4: Estructuras de datos en bajo nivel (imperativo/destructivo)

Linked lists (destructivas). Árboles con punteros. Iteración en listas y árboles (puros). Binary Heaps con arrays. Hashing.

Unidad 5: Temas adicionales

Sorting. Algoritmos de ordenamiento: insertion sort, selection sort, merge sort y quick sort. Implementaciones en funcional y en imperativo.

Bibliografía obligatoria:

- Chris Okasaki, Purely Functional Data Structures. Cambridge University Press, 2009.
- Lipovaca, M. Learn you a Haskell for great good!: A beginner's guide. No starch press. 1st Edition. 2011
- Barnett, G., & Del Tongo, L. Data Structures and Algorithms: Annotated Reference with Examples. DotNetSlackers. 2008

- Reema Thareja. Data Structures Using C. Oxford. 2014.
- Mark Allen Weiss , Data Structures and Algorithm in C. Addison-Wesley, 1997 (2da edición).

Bibliografía de consulta:

- Thomas Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein , Introduction to Algorithms. MIT Press, 2009.
- Peter A. Fejer y Dan A. Simovici, Mathematical Foundations of Computer Science. Vol.I: Sets, Relations, and Induction. Springer Verlag, 1991..

Organización de las clases:

Se realizarán clases teóricas (una vez por semana) y prácticas de programación en papel y computadora (dos veces por semana).

TP N°1:

Los objetivos del trabajo son: familiarizarse con el lenguaje Haskell; aprender a definir tipos algebraicos, y funciones a través del uso de pattern matching, polimorfismo paramétrico (registros, tipos algebraicos como Maybe, Either, etc.).

TP N°2:

Los objetivos del trabajo son: aprender recursión sobre listas; aprender a definir funciones sobre listas.

TP N°3:

Los objetivos del trabajo son: aprender a definir tipos algebraicos arbóreos y funciones sobre los mismos utilizando recursión.

TP N°4:

Los objetivos del trabajo son: aprender a definir tipos abstractos simples, definir interfaces, aprender a calcular costos sobre funciones y comprender el uso de tipos abstractos cómo mecanismo de reutilización de código.

TP N°5:

Los objetivos del trabajo son: aprender a definir distintas variantes de pilas y colas como tipos abstractos y aprender algunos algoritmos que se sustentan sobre dichas estructuras.

TP N°6:

Los objetivos del trabajo son: aprender a definir diccionarios utilizando listas, así como su uso en diversos algoritmos.

TP N°7:

Los objetivos del trabajo son: aprender a definir diccionarios mediante árboles de búsqueda binaria, y aprender a definir algoritmos utilizando árboles de búsqueda en general. Además, se verá cómo mejorar la eficiencia a través de la definición de AVLs.

TP N°8:

Los objetivos del trabajo son: aprender a colas de prioridad a través de binary heaps, así como algoritmos donde esta estructura está involucradas, como heapsort.

TP N°9:

Los objetivos del trabajo son: introducir el uso C++, aprender a resolver problemas utilizando estructuras, arrays y punteros. Modelar registros como tipos abstractos a través de módulos.

TP N°10:

Los objetivos del trabajo son: aprender a definir listas enlazadas utilizando punteros. Aprender ciertas mejoras sobre la definición de estas listas, mejorando la eficiencia de toda su interfaz. Aprender a definir iteradores sobre las mismas.

TP N°11:

Los objetivos del trabajo son: aprender a definir algoritmos imperativos sobre árboles binarios utilizando DFS y BFS. Aprender a definir árboles de búsqueda

binaria y heaps utilizando arrays.

TP N°12:

Los objetivos del trabajo son: aprender a definir diccionarios utilizando tablas de hash. Aprender ciertos algoritmos clásicos de ordenamiento, principalmente sobre arrays.

Modalidad de evaluación:

Los mecanismos de evaluación en modalidad presencial de esta asignatura están reglamentados según los siguientes artículos del Régimen de estudios de la UNQ (Res. CS 201/18).

CRONOGRAMA TENTATIVO

Semana	Tema/unidad	Actividad*				Evaluación
		Teórico	Práctico			
			Res Prob	Lab.	Otros Especificar	
1	Presentación de Haskell y Hugs, Tipos Algebraicos, Pattern Matching	x	x	x		
2	Listas y recursión sobre listas.	x	x	x		
3	Árboles Binarios y recursión sobre árboles.	x	x	x		
4	Repaso	x	x	x		
5	Evaluación de Tipos Algebraicos.					x
6	Tipos Abstractos de Datos. Invariantes de Representación. Implementaciones de Colas, Pilas, Conjuntos	x	x	x		
7	Implementaciones simples de diccionarios (Maps). Eficiencia.	x	x	x		
8	Implementaciones logarítmicas de diccionarios (Maps). BSTs, AVLs.	x	x	x		
9	Colas con prioridad y Heaps.	x	x	x		
10	1er Parcial: Tipos Abstractos de Datos					x
11	Modelo de memoria. Representación de datos en memoria estática. Registros.	x	x	x		
12	Memoria dinámica. Punteros, Pasaje por Referencia. Representación de datos	x	x	x		

	usando memoria dinámica.					
13	Listas destructivas y funcionales, recorridos iterativos y recursivos.	x	x	x		
14	Arrays, Listas con iteradores, recorridos con iteradores. Recuperatorio 1er parcial.	x	x	x		x
15	Arboles Binarios destructivos, recorridos iterativos de árboles, DFS, BFS	x	x	x		
16	Heaps	x	x	x		
15	2do Parcial: Modelo Imperativo					x
17	Hashing. Sorting	x	x	x		
18	Recuperatorio 2do parcial.					x
19	Integrador					x