



National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS210. Algorithms and Data Structures (Mandatory)

2. GENERAL INFORMATION

2.1 Course	: CS210. Algorithms and Data Structures
2.2 Semester	: 4 th Semester.
2.3 Credits	: 4
2.4 Hours	: 2 HT; 4 HP;
2.5 Duration of the period	: 16 weeks
2.6 Type of course	: Mandatory
2.7 Learning modality	: Face to face
2.8 Prerequisites	: CS113. Objects-oriented programming II. (3 rd Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This course provides a comprehensive study of fundamental data structures and algorithms with emphasis on implementation using advanced C++ features. Building on prior C++ knowledge, students will implement generic data structures using templates and traits, ensuring type flexibility and performance. The course covers linear structures (vectors, various linked lists), tree structures (binary, AVL, B-trees), heaps, hash tables, tries, and concurrent data structures. Special focus is placed on modern C++ techniques including variadic templates for generic operations and concurrent programming for thread-safe data structures.

5. GOALS

- Implement generic data structures using templates and traits.
- Master advanced tree structures and balanced trees.
- Understand and implement concurrent data structures.
- Apply variadic templates for generic operations on containers.
- Analyze algorithmic complexity of all implemented structures.
- Design efficient data structures for specific problem constraints.

6. COMPETENCES

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

7. TOPICS

Unit 1: Advanced Foundation for Data Structures (8 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	Learning Outcomes
• Template metaprogramming review and advanced traits. • Type traits and SFINAE for conditional compilation. • Variadic templates and parameter packs. • Perfect forwarding and universal references. • Lambda expressions with generic captures. • Policy-based design and CRTP (Curiously Recurring Template Pattern). • Concept-based constraints (C++20 concepts if applicable).	• Design type traits for customizing data structure behavior [Usar]. • Implement variadic templates for generic container operations [Usar]. • Apply policy-based design to create configurable data structures [Evaluar]. • Use perfect forwarding to implement efficient constructors [Usar].
Readings : [Stroustrup2013], [Vandevorde2017]	

Unit 2: Foundational Data Structures (8 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	Learning Outcomes
• Abstract Data Type (ADT) and operations on an ADT: – Dictionary operations (insert, delete, find) • Arrays: – Numeric vs non-numeric, character strings – Single (vector) vs multidimensional (matrix) • Records/Structs/Tuples and Objects • Linked lists (for historical reasons): – Single vs Double and Linear vs Circular • Stacks • Queues and dequeues: – Heap-based priority queue • Hash tables/maps: – Collision resolution and complexity (e.g., probing, chaining, rehash) • Trees: – Binary, n-ary, and search trees – Balanced (e.g., AVL, Red-Black, Heap) • Sets	• For each ADT/Data-Structure in this unit: – Explain its definition, properties, representation(s), and associated ADT operations. – Explain step-by-step how the ADT operations associated with the data structure transform it. [Usar] • For each algorithm in this unit explain step-by-step how the algorithm operates [Usar] • Implement vector with customizable allocation policy using templates [Usar]. • Design generic linked list nodes with trait-based memory management [Usar]. • Compare time/space complexity of different linear structures [Evaluar]. • Apply variadic templates to initialize containers with multiple elements [Usar].
Readings : [Cormen2009], [Stroustrup2013]	

Unit 3: Foundational Data Structures (8 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	Learning Outcomes
• Trees: – Binary, n-ary, and search trees – Balanced (e.g., AVL, Red-Black, Heap) • Binary tree template implementation with iterator support. • Binary search tree with trait-based comparison operators. • Tree traversal algorithms (iterative and recursive). • Expression trees and their applications. • Template-based node structures with configurable payload.	• Given requirements for a problem, develop multiple solutions using various data structures and algorithms. Subsequently, evaluate the suitability, strengths, and weaknesses selecting an approach that best satisfies the requirements [Evaluar] • Explain factors beyond computational efficiency that influence the choice of algorithms, such as programming time, maintainability, and the use of application-specific patterns in the input data [Familiarizarse] • Implement generic binary tree with configurable traversal order [Usar]. • Design expression tree evaluator using visitor pattern [Usar]. • Analyze tree operation complexities for different balancing strategies [Evaluar].
Readings : [Cormen2009], [Knuth1997]	

Unit 4: Balanced and Multi-way Trees (8 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	Learning Outcomes
• AVL tree implementation with template-based balancing trait. • Red-black tree properties and operations. • B-Tree template implementation for disk-based storage. • 2-3 Tree and 2-3-4 Tree structures. • Digital trie (prefix tree) implementation. • Suffix trees and their applications in string processing. • Node splitting/merging strategies for B-trees.	• Implement self-balancing trees using policy-based design [Usar]. • Design B-tree for database index applications [Usar]. • Evaluate trade-offs between different balanced tree structures [Evaluar]. • Implement trie for efficient string prefix matching [Usar].
Readings : [Cormen2009], [Knuth1997], [Sedgewick2011]	

Unit 5: Foundational Data Structures (8 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	Learning Outcomes
• Trees: – Binary, n-ary, and search trees – Balanced (e.g., AVL, Red-Black, Heap) • Binary heap implementation using array representation. • Min-heap and max-heap variants. • Heapify algorithms and complexity analysis. • Priority queue ADT implementation. • Binomial heap and Fibonacci heap overview. • Heap applications: Dijkstra’s algorithm, heap sort.	• Explain the heap property and the use of heaps as an implementation of a priority queue [Familiarizarse] • Implement generic heap with configurable comparison trait [Usar]. • Design priority queue supporting multiple priority update strategies [Usar]. • Analyze heap operations complexity for different heap types [Evaluar]. • Apply heap in graph algorithm implementations [Usar].
Readings : [Cormen2009], [Sedgewick2011]	

Unit 6: Foundational Data Structures (8 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	Learning Outcomes
• Hash tables/maps: – Collision resolution and complexity (e.g., probing, chaining, rehash) • Hash function design and properties. • Separate chaining implementation with linked lists. • Open addressing: linear probing, quadratic probing, double hashing. • Perfect hashing and minimal perfect hashing. • Cuckoo hashing and hopscotch hashing. • Hash table resizing strategies and load factor management.	• Explain how collision avoidance and collision resolution is handled in hash tables [Familiarizarse] • Design generic hash table with configurable collision resolution [Usar]. • Implement custom hash functions for user-defined types [Usar]. • Evaluate performance of different hashing schemes [Evaluar]. • Apply hash tables in compiler symbol table implementation [Usar].
Readings : [Cormen2009], [Knuth1997]	

Unit 7: Concurrent Data Structures (8 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	Learning Outcomes
• Thread-safe data structure design principles. • Lock-based concurrent linked lists. • Lock-free and wait-free algorithms. • Concurrent hash tables with fine-grained locking. • Atomic operations and memory ordering. • Concurrent queues (bounded and unbounded). • Concurrent stacks and elimination techniques.	• Design thread-safe data structures using mutex and lock guards [Usar]. • Implement lock-free linked list using atomic operations [Usar]. • Analyze performance of concurrent vs sequential data structures [Evaluar]. • Apply concurrent queues in producer-consumer patterns [Usar].
Readings : [Herlihy2012], [Williams2019]	

Unit 8: Specialized Structures and Applications (8 hours)	
Competences Expected: AG-C08,AG-C09	
Topics	Learning Outcomes
• Persistent data structures. • Disjoint-set union (union-find) with path compression. • Bloom filters and their probabilistic properties. • Skip lists for probabilistic balanced search. • Segment trees and Fenwick trees for range queries. • Spatial data structures: k-d trees, quadtrees. • Cache-oblivious data structures.	• Implement persistent data structures with structural sharing [Usar]. • Design union-find with both union by rank and path compression [Usar]. • Apply specialized structures to solve domain-specific problems [Evaluar]. • Evaluate space-time tradeoffs in probabilistic data structures [Evaluar].
Readings : [Okasaki1999], [Sedgewick2011]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:

In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:

Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE : Partial Exam.
- FE : Final Exam.
- P : Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY