National University of Engineering (UNI)
School of Computer Science
Syllabus 2026-I

1. COURSE

CS113. Objects-oriented programming II (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS113. Objects-oriented programming II
2.2 Semester	:	3^{rd} 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 Prerrequisites	:	CS112. Objects-oriented programming I. (2^{nd} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

This is the third course in the sequence of introductory courses in computer science. This course explores advanced C++17/C++20 features focusing on high-performance systems development. Key topics include template metaprogramming, move semantics, perfect forwarding, RAII optimization, multiple/virtual inheritance patterns, concurrency with std::thread and async/await, lock-free programming, modern C++ paradigms such as CRTP and expression templates, and interfacing C++ with other languages. The course prepares students for game engines, high-performance computing, and embedded development where C++ dominates.

5. GOALS

- Master advanced C++17/C++20 features including templates and metaprogramming
- Develop skills in concurrent and systems programming
- Understand performance optimization techniques
- Prepare for high-performance computing applications
- Master template metaprogramming and compile-time computation

6. COMPETENCES

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

AG-C12) Uses systemic approaches to select, develop, apply, integrate, and manage secure computing technologies to achieve user objectives. (Assessment)

7. TOPICS

Unit 1: Programación Orientada a Objetos II: Encapsulación, Subtipado y Reflexión (8 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Clases de colecciones, iteradores y otros componentes comunes de biblioteca. • Associative containers (std::set, std::map, unordered variants) • Container adapters (stack, queue, priority_queue) • Advanced STL algorithms • Functors and predicates • Iterator categories and traits	• Usar clases de colecciones e iteradores efectivamente para resolver un problema [Usar] • Definir y usar iteradores y otras operaciones en agregados, incluyendo operaciones que toman funciones como argumentos, en múltiples lenguajes de programación, seleccionando los idiomas más naturales para cada lenguaje [Evaluar] • Comparar y contrastar los beneficios y costos/impacto de usar herencia (subclases) y composición (específicamente, cómo basar la composición en funciones de orden superior) [Usar] • Implement custom allocators for STL containers [Usar].
Readings : [**Stroustrup2013**], [**Josuttis2019**], [**Vandevoorde2017**]	

Unit 2: Programación Funcional (12 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Expresiones lambda y evaluación: – Enlace (binding) de variables y reglas de ámbito (scope). – Paso de parámetros. – Expresiones lambda anidadas y orden de reducción. • Clausuras (*closures*) de funciones (funciones que usan variables en el entorno léxico circundante): – Significado y definición básica: crear clausuras en tiempo de ejecución capturando el entorno. – Idiomas canónicos: devoluciones de llamada (*callbacks*), argumentos para iteradores, código reutilizable mediante argumentos de funciones. – Usar una clausura para encapsular datos en su entorno. – Evaluación diferida (*lazy*) vs ansiosa (*eager*). • Template metaprogramming techniques • SFINAE (Substitution Failure Is Not An Error) • Perfect forwarding and universal references • Variadic templates • Compile-time computation with constexpr	• Desarrollar funciones útiles que tomen y devuelvan otras funciones [Usar] • Interpretar correctamente las variables y el ámbito léxico en un programa que usa clausuras de funciones [Evaluar] • Usar mecanismos de encapsulación funcional como clausuras e interfaces modulares [Usar] • Apply SFINAE for template selection and overload resolution [Usar]. • Implement perfect forwarding for efficient argument passing [Usar].
Readings : [**Stroustrup2013**], [**Josuttis2019**], [**Gregor2017**]	

Unit 3: Gestión de Memoria (8 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Asignación y acceso de bajo nivel a estructuras de datos de alto nivel como tipos de datos básicos, arreglos n-dimensionales, vectores, registros y objetos • Retorno de procedimiento como mecanismo de desasignación automática para elementos de datos locales en la pila • Gestión manual de memoria: asignar, desasignar y reutilizar memoria del montón • Gestión automática de memoria: recolección de basura (garbage collection) como técnica automatizada usando la noción de alcanzabilidad (reachability) • Lvalues and rvalues in C++11/14/17/20 • Move constructors and move assignment operators • Rvalue references and reference collapsing • Resource Acquisition Is Initialization (RAII) pattern • Smart pointer customization • Memory leak detection and prevention	• Explicar por qué ocurren las fugas de memoria (memory leaks) y los problemas de punteros colgantes (dangling pointers), y qué puede hacer un programador para evitarlos/corregirlos [Usar] • Explicar cómo las implementaciones de lenguajes de programación típicamente organizan la memoria en secciones de datos globales, texto, montón y pila, y cómo características como recursión y gestión de memoria se mapean a este modelo de memoria [Evaluar] • Explicar cómo se ejecuta una construcción central del lenguaje, como abstracciones de datos y abstracciones de control [Evaluar] • Implement move semantics for resource optimization [Usar]. • Design RAII classes for exception-safe resource management [Usar].
Readings : [**Meyers2014**], [**Stroustrup2013**]	

Unit 4: Computación Paralela y Distribuida (12 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Seguridad (safety) y vivacidad (liveness): – Condiciones de carrera (race conditions) – Dependencias/precondiciones – Modelos de fallos (fault models) – Terminación • Modelos de programación: – Modelo de actores (*actor models*) – Modelos procedurales y reactivos – Modelos de programación síncrona/asíncrona – Paralelismo de datos (*data parallelism*) • Comunicación y coordinación: – Mutexes – Paso de mensajes (*message-passing*) – Memoria compartida (*shared memory*) – Cobegin-coend – Monitores – Canales (*channels*) – Hilos (*threads*) – Guardas (*guards*) • std::thread and thread management • Mutexes, locks, and condition variables • Atomic operations and lock-free programming • std::async and futures • Thread-safe design patterns • Data races and deadlock prevention	• Implementar programas concurrentes correctos usando múltiples modelos de programación, como memoria compartida, actores, futuros, construcciones de sincronización y primitivas de paralelismo de datos [Usar] • Usar construcciones de sincronización como métodos monitorizados/sincronizados en un programa simple [Usar] • Usar un modelo de paso de mensajes para analizar un protocolo de comunicación [Evaluar] • Implement lock-free data structures using atomics [Usar]. • Analyze thread-safety of concurrent algorithms [Evaluar].
Readings : [**Williams2019**], [**Stroustrup2013**]	

Unit 5: Construcciones de Programación Avanzadas (8 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Abstracciones orientadas a objetos: herencia múltiple, mixins, rasgos (traits), multimétodos (multimethods) • Metaprogramación: macros, programación generativa, desarrollo basado en modelos (model-based development) • Multiple inheritance and virtual inheritance • CRTP (Curiously Recurring Template Pattern) • Expression templates • Type traits and compile-time introspection • Policy-based design • Interface design for libraries	• Usar varias construcciones e idiomas de programación avanzados correctamente [Usar] • Discutir cómo varias construcciones de programación avanzadas tienen como objetivo mejorar la estructura del programa, la calidad del software y la productividad del programador [Evaluar] • Discutir cómo varias construcciones de programación avanzadas interactúan con la definición e implementación de otras características del lenguaje [Evaluar] • Implement CRTP for static polymorphism [Usar]. • Design extensible library interfaces using policy-based design [Usar].
Readings : [**Stroustrup2013**], [**Gregor2017**]	

Unit 6: Modelo de Ejecución y Memoria de Sistemas (8 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Representación en tiempo de ejecución de abstracciones de datos como variables, arreglos, vectores, registros, elementos de datos basados en punteros como listas enlazadas y árboles, y objetos • Estructuras de datos para traducción, ejecución, traducción y movilidad de código como pila, montón (heap), alias (compartición usando punteros), secuencia indexada y cadena • Interfacing C++ with other languages (FFI) calls and OS interaction • Memory-mapped I/O • Performance profiling and benchmarking • Optimization techniques for HPC	• Explicar cómo se ejecuta una construcción central del lenguaje, como abstracciones de datos y abstracciones de control [Usar] • Explicar cómo las implementaciones de lenguajes de programación típicamente organizan la memoria en secciones de datos globales, texto, montón y pila, y cómo características como recursión y gestión de memoria se mapean a este modelo de memoria [Evaluar] • Explicar por qué ocurren las fugas de memoria (memory leaks) y los problemas de punteros colgantes (dangling pointers), y qué puede hacer un programador para evitarlos/corregirlos [Usar] • Profile and optimize C++ programs for performance [Usar]. • Interface C++ code with C libraries [Usar].
Readings : [**Stroustrup2013**], [**Williams2019**]	

Unit 7: Construcciones de Programación Avanzadas (6 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Abstracciones orientadas a objetos: herencia múltiple, mixins, rasgos (traits), multimétodos (multimethods) • Creational, structural, and behavioral patterns • Thread-safe singleton patterns • Modern C++ design idioms • Dependency injection and inversion of control	• Usar varias construcciones e idiomas de programación avanzados correctamente [Usar] • Discutir cómo varias construcciones de programación avanzadas tienen como objetivo mejorar la estructura del programa, la calidad del software y la productividad del programador [Evaluar] • Discutir cómo varias construcciones de programación avanzadas interactúan con la definición e implementación de otras características del lenguaje [Familiarizarse] • Implement thread-safe design patterns [Usar]. • Evaluate architectural trade-offs in C++ systems [Evaluar].
Readings : [**Gamma1994**], [**Stroustrup2013**]	

Unit 8: Construcciones de Programación Avanzadas (6 hours)	
Competences Expected: AG-C09,AG-C12	
Topics	**Learning Outcomes**
• Abstracciones de control: manejo de excepciones, continuaciones (continuations), mónadas (monads). • Compiler optimizations and flags • Benchmarking methodologies • Cache-friendly data structures • Branch prediction and alignment • Reducing compilation time integration for performance testing	• Usar varias construcciones e idiomas de programación avanzados correctamente [Usar] • Discutir cómo varias construcciones de programación avanzadas tienen como objetivo mejorar la estructura del programa, la calidad del software y la productividad del programador [Evaluar] • Discutir cómo varias construcciones de programación avanzadas interactúan con la definición e implementación de otras características del lenguaje [Familiarizarse] • Apply optimization techniques to improve program performance [Usar]. • Interpret benchmark results to guide optimization decisions [Usar].
Readings : [**Meyers2014**], [**Stroustrup2013**]	

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

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY