Universidad Nacional de Ingeniería (UNI)
Escuela Profesional de
Ciencia de la Computación
Sílabo 2026-I

1. CURSO

CS113. Programación Orientada a Objetos II (Obligatorio)

2. INFORMACIÓN GENERAL

2.1 Curso	:	CS113. Programación Orientada a Objetos II
2.2 Semestre	:	3^{er} Semestre.
2.3 Créditos	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duración del periodo	:	16 semanas
2.6 Condición	:	Obligatorio
2.7 Modalidad de aprendizaje	:	Presencial
2.8 Prerrequisitos	:	CS112. Programación Orientada a Objetos I. (2^{do} Sem)

3. PROFESORES

Atención previa coordinación con el profesor

4. INTRODUCCIÓN AL CURSO

Este es el tercer curso en la secuencia de cursos introductorios en ciencia de la computación. Este curso explora características avanzadas de C++17/C++20 centrándose en el desarrollo de sistemas de alto rendimiento. Los temas clave incluyen metaprogramación con plantillas, semántica de movimiento, reenvío perfecto, optimización RAII, patrones de herencia múltiple/virtual, concurrencia con std::thread y async/await, programación sin bloqueo, paradigmas modernos de C++ como CRTP y plantillas de expresión, e interfaz de C++ con otros lenguajes. El curso prepara a los estudiantes para motores de juego, computación de alto rendimiento y desarrollo embebido donde C++ es dominante.

5. OBJETIVOS

- Dominar características avanzadas de C++17/C++20, incluyendo plantillas y metaprogramación.
- Desarrollar habilidades en programación concurrente y de sistemas.
- Comprender técnicas de optimización de rendimiento.
- Prepararse para aplicaciones de computación de alto rendimiento.
- Dominar la metaprogramación con plantillas y el cómputo en tiempo de compilación.

6. RESULTADOS DEL ESTUDIANTE

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usage)

AG-C12) Usa enfoques sistémicos para seleccionar, desarrollar, aplica, integrar y administrar tecnologías seguras de computación para lograr los objetivos del usuario. (Assessment)

7. TEMAS

Unidad 1: Programación Orientada a Objetos II: Encapsulación, Subtipado y Reflexión (8 horas)	
Resultados esperados: AG-C09,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Clases de colecciones, iteradores y otros componentes comunes de biblioteca. • Contenedores asociativos (std::set, std::map, variantes no ordenadas). • Adaptadores de contenedor (stack, queue, priority_queue). • Algoritmos avanzados de STL. • Funtores y predicados. • Categorías de iteradores y rasgos.	• 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] • Implementar asignadores personalizados para contenedores STL [Usar].
Lecturas : [Str13], [Jos19], [VJG17]	

Unidad 2: Programación Funcional (12 horas)	
Resultados esperados: AG-C09,AG-C12	
Temas	**Aprendizaje esperado (*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*). • Técnicas de metaprogramación con plantillas. • SFINAE (Fallo de Sustitución No es un Error). • Reenvío perfecto y referencias universales. • Plantillas variádicas. • Cómputo en tiempo de compilación con 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] • Aplicar SFINAE para selección de plantillas y resolución de sobrecarga [Usar]. • Implementar reenvío perfecto para paso eficiente de argumentos [Usar].
Lecturas : [Str13], [Jos19], [Gre17]	

Unidad 3: Gestión de Memoria (8 horas)	
Resultados esperados: AG-C09,AG-C12	
Temas	**Aprendizaje esperado (*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 y rvalues en C++11/14/17/20. • Constructores de movimiento y operadores de asignación de movimiento. • Referencias a valor-r y colapso de referencias. • Patrón RAII (Adquisición de Recursos es Inicialización). • Personalización de punteros inteligentes. • Detección y prevención de fugas de memoria.	• 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] • Implementar semántica de movimiento para optimización de recursos [Usar]. • Diseñar clases RAII para gestión de recursos segura ante excepciones [Usar].
Lecturas : [Mey14], [Str13]	

Unidad 4: Computación Paralela y Distribuida (12 horas)	
Resultados esperados: AG-C09,AG-C12	
Temas	**Aprendizaje esperado (*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 y gestión de hilos. • Mutexes, candados y variables de condición. • Operaciones atómicas y programación sin bloqueo. • std::async y futures. • Patrones de diseño seguros para hilos. • Carreras de datos y prevención de interbloqueos.	• 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] • Implementar estructuras de datos sin bloqueo usando operaciones atómicas [Usar]. • Analizar la seguridad en hilos de algoritmos concurrentes [Evaluar].
Lecturas : [Wil19], [Str13]	

Unidad 5: Construcciones de Programación Avanzadas (8 horas)	
Resultados esperados: AG-C09,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Abstracciones orientadas a objetos: herencia múltiple, mixins, rasgos (traits), multimétodos (multimethods) • Metaprogramación: macros, programación generativa, desarrollo basado en modelos (model-based development) • Herencia múltiple y herencia virtual. • Patrón CRTP (Patrón de Plantilla Curiosamente Recursivo). • Plantillas de expresión. • Rasgos de tipos e introspección en tiempo de compilación. • Diseño basado en políticas. • Diseño de interfaz para bibliotecas.	• 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] • Implementar CRTP para polimorfismo estático [Usar]. • Diseñar interfaces de biblioteca extensibles usando diseño basado en políticas [Usar].
Lecturas : [Str13], [Gre17]	

Unidad 6: Modelo de Ejecución y Memoria de Sistemas (8 horas)	
Resultados esperados: AG-C09,AG-C12	
Temas	**Aprendizaje esperado (*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 • Interfaz de C++ con otros lenguajes (FFI). • Llamadas al sistema e interacción con el SO. • E/S mapeada en memoria. • Perfilado de rendimiento y evaluación comparativa. • Técnicas de optimización para 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] • Perfilar y optimizar programas C++ para rendimiento [Usar]. • Interfazar código C++ con bibliotecas C [Usar].
Lecturas : [Str13], [Wil19]	

Unidad 7: Construcciones de Programación Avanzadas (6 horas)	
Resultados esperados: AG-C09,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Abstracciones orientadas a objetos: herencia múltiple, mixins, rasgos (traits), multimétodos (multimethods) • Patrones creacionales, estructurales y de comportamiento. • Patrones singleton seguros para hilos. • Modismos de diseño modernos en C++. • Inyección de dependencias e inversión de 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] • Implementar patrones de diseño seguros para hilos [Usar]. • Evaluar compensaciones arquitectónicas en sistemas C++ [Evaluar].
Lecturas : [Gam+94], [Str13]	

Unidad 8: Construcciones de Programación Avanzadas (6 horas)	
Resultados esperados: AG-C09,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Abstracciones de control: manejo de excepciones, continuaciones (continuations), mónadas (monads). • Optimizaciones del compilador y banderas. • Metodologías de evaluación comparativa. • Estructuras de datos amigables con la caché. • Predicción de bifurcación y alineación. • Reducción del tiempo de compilación. • Integración continua para pruebas de rendimiento.	• 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] • Aplicar técnicas de optimización para mejorar el rendimiento del programa [Usar]. • Interpretar resultados de evaluación comparativa para guiar decisiones de optimización [Usar].
Lecturas : [Mey14], [Str13]	

8. PLAN DE TRABAJO

8.1 Metodología

Se fomenta la participación individual y en equipo para exponer sus ideas, motivándolos con puntos adicionales en las diferentes etapas de la evaluación del curso.

8.2 Sesiones Teóricas

Las sesiones de teoría se llevan a cabo en clases magistrales donde se realizarán actividades que propicien un aprendizaje activo, con dinámicas que permitan a los estudiantes interiorizar los conceptos.

8.3 Sesiones Prácticas

Las sesiones prácticas se llevan en clase donde se desarrollan una serie de ejercicios y/o conceptos prácticos mediante planteamiento de problemas, la resolución de problemas, ejercicios puntuales y/o en contextos aplicativos.

9. SISTEMA DE EVALUACIÓN

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

10. BIBLIOGRAFÍA BÁSICA

[Gam+94] Erich Gamma et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. 1st. Reading, MA: Addison-Wesley Professional, 1994.

[Str13] Bjarne Stroustrup. *The C++ Programming Language*. 4th. Upper Saddle River, NJ: Addison-Wesley Professional, 2013.

[Mey14] Scott Meyers. *Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14*. 1st. Sebastopol, CA: O'Reilly Media, 2014.

[Gre17] Douglas Gregor. *Advanced C++: Master the Technique of Template Metaprogramming*. 1st. Upper Saddle River, NJ: Addison-Wesley Professional, 2017.

[VJG17] David Vandevoorde, Nicolai M. Josuttis, and Douglas Gregor. *C++ Templates: The Complete Guide*. 2nd. Upper Saddle River, NJ: Addison-Wesley Professional, 2017.

[Jos19] Nicolai M. Josuttis. *C++17: The Complete Guide*. 1st. Indianapolis, IN: Pearson Education, 2019.

[Wil19] Anthony Williams. *C++ Concurrency in Action*. 2nd. Shelter Island, NY: Manning Publications, 2019.