



Universidad Nacional de Ingeniería (UNI)

Escuela Profesional de
Ciencia de la Computación
Sílabo 2024-II

1. CURSO

CS112. Ciencia de la Computación I (Obligatorio)

2. INFORMACIÓN GENERAL

2.1 Curso	: CS112. Ciencia de la Computación I
2.2 Semestre	: 2 ^{do} Semestre.
2.3 Créditos	: 5
2.4 horas	: 2 HT; 6 HP;
2.5 Duración del periodo	: 16 semanas
2.6 Condición	: Obligatorio
2.7 Modalidad de aprendizaje	: Presencial
2.8 Prerrequisitos	: CS111. Introducción a la Programación. (1 ^{er} Sem)

3. PROFESORES

Atención previa coordinación con el profesor

4. INTRODUCCIÓN AL CURSO

Este es el segundo curso en la secuencia de los cursos introductorios a la Ciencia de la Computación. El curso introducirá a los participantes en los diversos temas del área de computación como: algoritmos, estructuras de datos, ingeniería del software, etc.

5. OBJETIVOS

- Introducir al alumno a los fundamentos del paradigma de orientación a objetos, permitiendo asimilar los conceptos necesarios para desarrollar sistemas de información.

6. RESULTADOS DEL ESTUDIANTE

- 1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Assessment)
- 2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline. (Assessment)
- 5) Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline. (Familiarity)
- 6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Usage)

7. TEMAS

Unidad 1: Visión general de los lenguajes de programación (1 horas)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
- Breve repaso de los paradigmas de programación. - Comparación entre programación funcional y programación imperativa. - Historia de los lenguajes de programación (énfasis en C y C++).	Discutir el contexto histórico de los paradigmas de diversos lenguajes de programación [Familiarizarse]
Lecturas : [Str13], [Jos19], [Dei17]	

Unidad 2: Máquinas virtuales (2 horas)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
- El concepto de máquina virtual. - Tipos de virtualización (incluyendo Hardware / Software, OS, Servidor, Servicio, Red) . - Lenguajes intermedios.	- Explicar el concepto de memoria virtual y la forma cómo se realiza en hardware y software [Familiarizarse] - Diferenciar emulación y el aislamiento [Familiarizarse] - Evaluar virtualización de compensaciones [Evaluar]
Lecturas : [Str13], [Jos19], [Dei17]	

Unidad 3: Sistemas de tipos básicos (6 horas)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
• Sistemas de tipos en lenguajes de programación. • Modelos de declaración (enlace, visibilidad, alcance y tiempo de vida). • Resumen de la verificación de tipos.	• Tanto para tipo primitivo y un tipo compuesto, describir de manera informal los valores que tiene dicho tipo [Familiarizarse] • Para un lenguaje con sistema de tipos estático, describir las operaciones que están prohibidas de forma estática, como pasar el tipo incorrecto de valor a una función o método [Familiarizarse] • Describir ejemplos de errores de programa detectadas por un sistema de tipos [Familiarizarse] • Para múltiples lenguajes de programación, identificar propiedades de un programa con verificación estática y propiedades de un programa con verificación dinámica [Usar] • Dar un ejemplo de un programa que no verifique tipos en un lenguaje particular y sin embargo no tenga error cuando es ejecutado [Familiarizarse] • Usar tipos y mensajes de error de tipos para escribir y depurar programas [Usar] • Explicar como las reglas de tipificación definen el conjunto de operaciones que legales para un tipo [Familiarizarse] • Escribir las reglas de tipo que rigen el uso de un particular tipo compuesto [Usar] • Explicar por qué indecidibilidad requiere sistemas de tipo para conservadoramente aproximar el comportamiento de un programa [Familiarizarse] • Definir y usar piezas de programas (tales como, funciones, clases, métodos) que usan tipos genéricos, incluyendo para colecciones [Usar] • Discutir las diferencias entre, genéricos (<i>generics</i>), subtipo y sobrecarga [Familiarizarse] • Explicar múltiples beneficios y limitaciones de tipificación estática en escritura, mantenimiento y depuración de un software [Familiarizarse]
Lecturas : [Str13], [Jos19], [Dei17]	

Unidad 4: Conceptos Fundamentales de Programación (6 horas)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
• Diseño orientado a objetos: – Descomposición en objetos que almacenan estados y poseen comportamiento – Diseño basado en jerarquía de clases para modelamiento • Variables y tipos de datos. • Expresiones y operadores. • Sentencias condicionales.	• Analiza y explica el comportamiento de programas simples que involucran estructuras fundamentales de programación variables, expresiones, asignaciones, E/S, estructuras de control, funciones, paso de parámetros, y recursividad [Evaluar] • Identifica y describe el uso de tipos de datos primitivos [Familiarizarse] • Escribe programas que usan tipos de datos primitivos [Usar] • Modifica y expande programas cortos que usen estructuras de control condicionales e iterativas así como funciones [Usar] • Diseña, implementa, prueba, y depura un programa que usa cada una de las siguientes estructuras de datos fundamentales: cálculos básicos, E/S simple, condicional estándar y estructuras iterativas, definición de funciones, y paso de parámetros [Usar] • Elige estructuras de condición y repetición adecuadas para una tarea de programación dada [Evaluar]
Lecturas : [Str13], [Jos19], [Dei17]	

Unidad 5: Funciones (6 horas)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
• Paso de funciones y parámetros. • Paso de parámetros. • Sobrecarga de funciones. • Fundamentos de la recursividad. • Plantillas de funciones.	• Diseña, implementa, prueba, y depura un programa que usa cada una de las siguientes estructuras de datos fundamentales: cálculos básicos, E/S simple, condicional estándar y estructuras iterativas, definición de funciones, y paso de parámetros [Usar] • Entiende y aplica el concepto de paso de parámetros a una función, tanto por valor como por referencia.[Usar] • Identifica y aplica el concepto de sobrecarga de funciones.[Usar] • Describe el concepto de recursividad y da ejemplos de su uso [Familiarizarse] • Diseña, implementa y aplica el concepto de plantillas asociado a la necesidad de crear funciones genéricas.[Usar]
Lecturas : [Str13], [Jos19], [Dei17]	

Unidad 6: Arreglos, punteros y gestión de memoria (8 horas)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
- Definición de arreglos. - Arreglos multidimensionales. - Fundamentos de punteros. - Gestión de memoria dinámica (new/delete, pila vs. heap). - Punteros inteligentes (unique_ptr, shared_ptr, weak_ptr). - Conceptos avanzados de punteros (punteros a punteros, punteros a funciones).	- Entiende e implementa arreglos unidimensionales. [Familiarizarse] - Diseña y aplica el concepto de arreglos multidimensionales. [Usar] - Entiende y aplica el concepto de referencias y punteros. [Familiarizarse] - Entiende, aplica y evalúa la relación entre punteros y arreglos. [Evaluar] - Entiende e implementa la gestión dinámica de la memoria. Diferenciando las regiones de memoria: heap y stack. [Evaluar] - Diseña, implementa y evalúa el concepto de puntero a puntero, puntero a función, entre otros conceptos. [Evaluar]
Lecturas : [Str13], [Jos19], [Dei17]	

Unidad 7: Manejo de punteros con arrays (5 horas)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
- Arrays como argumentos de una función. - Arrays de caracteres y punteros. - Punteros y Arrays de 2 dimensiones. - Punteros y arrays multidimensionales.	- Demostrar el uso de punteros con diferentes tipos de Arrays. [Usar] - Demostrar la disposición de un array en la memoria y como se manipula punteros dentro de esos espacios de memoria. [Usar] - Demostrar el uso de aritmética de punteros y arrays. [Usar]
Lecturas : [Str13], [Jos19], [Dei17]	

Unidad 8: Punteros y memoria dinámica (5 horas)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
• Punteros y memoria dinámica - stack vs heap. • Punteros como retorno de una función en C/C++. • Punteros a funciones en C/C++. • Punteros a funciones y callback. • Memory leak en C/C++.	• Mostrar la estructura de la memoria dentro de un programa y comprender cómo es que el compilador dispone elementos en el stack y en el heap.[Usar] • Demostrar el uso de las funciones y operadores de asignación y desasignación de memoria dinámica.[Usar] • Comprender las implicancias de retornar punteros desde funciones. [Usar] • Utilizar punteros a funciones como parámetros. [Usar] • Comprender la implicancia de uso de memoria dinámica y el memory leak. [Usar]
Lecturas : [Str13], [Jos19], [Dei17]	

Unidad 9: Punteros y clases (5 horas)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
• Punteros a miembros clase - atributos. • Punteros a miembros clase - métodos y llamadas a punteros a métodos. • Punteros a miembros clase - métodos static y llamadas a punteros a métodos static. • Punteros a clases - ejemplo con manejo de lista enlazada.	• Comprender el uso de punteros a diferentes elementos de una clase. [Usar] • Comprender el uso de punteros a miembros estáticos de una clase. [Usar] • Introducir en la estructura nodo y su uso en una estructura de datos simple. [Usar] • Introducir a las estructuras de datos, mostrando una implementación simple de listas enlazadas.[Usar]
Lecturas : [Str13], [Jos19], [Dei17]	

Unidad 10: Programación orientada a objetos (8 horas)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
• Diseño orientado a objetos: – Descomposición en objetos que almacenan estados y poseen comportamiento – Diseño basado en jerarquía de clases para modelamiento • Lenguajes orientados a objetos para la encapsulación: – privacidad y la visibilidad de miembros de la clase – Interfaces revelan único método de firmas – clases base abstractas • Definición de las categorías, campos, métodos y constructores. • Las subclases, herencia y método de alteración temporal. • Subtipificación: – Polimorfismo artículo Subtipo; upcasts implícitos en lenguajes con tipos. – Noción de reemplazo de comportamiento: los subtipos de actuar como supertipos. – Relación entre subtipos y la herencia. • Uso de colección de clases, iteradores, y otros componentes de la librería estándar. • Asignación dinámica: definición de método de llamada.	• Diseñar e implementar una clase [Usar] • Usar subclase para diseñar una jerarquía simple de clases que permita al código ser reusable por diferentes subclases [Usar] • Razonar correctamente sobre el flujo de control en un programa mediante el envío dinámico [Usar] • Comparar y contrastar (1) el enfoque procedur/funcional- definiendo una función por cada operación con el uso de la función proporcionando un caso por cada variación de dato - y (2) el enfoque orientado a objetos - definiendo una clase por cada variación de dato con la definición de la clase proporcionando un método por cada operación. Entender ambos enfoques como una definición de variaciones y operaciones de una matriz [Evaluar] • Explicar la relación entre la herencia orientada a objetos (código compartido y <i>overriding</i>) y subtipificación (la idea de un subtipo es ser utilizable en un contexto en el que espera al supertipo) [Familiarizarse] • Usar mecanismos de encapsulación orientada a objetos, tal como interfaces y miembros privados [Usar] • Definir y usar iteradores y otras operaciones sobre agregaciones, incluyendo operaciones que tienen funciones como argumentos, en múltiples lenguajes de programación, seleccionar la forma más natural por cada lenguaje [Usar]
Lecturas : [Str13], [Jos19], [Dei17]	

Unidad 11: Plantillas y STL (6 horas)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
• Plantillas de clases. • Conceptos básicos de la Biblioteca Estándar de Plantillas (STL) incluyendo: vector, list, stack, queue.	• Entiende los conceptos de plantillas en clases. [Familiarizarse] • Implementa y crea nuevos tipos de datos genéricos. [Usar] • Entiende las estructuras básicas de la STL. [Familiarizarse] • Usa las estructuras de datos básicas como: pila, cola, lista, vector contenidos en la STL. [Usar]
Lecturas : [Str13], [Jos19], [Dei17]	

Unidad 12: Sobrecarga de operadores (4 horas)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
Definición de sobrecarga de operadores.	- Entiende los conceptos de sobrecarga de operadores. [Familiarizarse] - Implementa la sobrecarga de operadores permitidos en el lenguaje de programación. [Usar]
Lecturas : [Str13], [Jos19], [Dei17]	

Unidad 13: Manejo de archivos (4 horas)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
Entrada y salida de archivos (I/O).	- Entiende los conceptos de manipulación de archivos. [Familiarizarse] - Crea programas de lectura y escrita en archivos. [Usar]
Lecturas : [Str13], [Jos19], [Dei17]	

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

[Str13] Bjarne Stroustrup. *The C++ Programming Language*. 4th. Addison-Wesley, 2013.

[Dei17] Deitel & Deitel. *C++17 - The Complete Guide*. 10th. Pearson, 2017.

[Jos19] Nicolai M. Josuttis. *C++17 - The Complete Guide*. 1st. 2019.