

1. CURSO

CS113. Ciencia de la Computación II (Obligatorio)

2. INFORMACIÓN GENERAL

2.1 Créditos	:	4
2.2 Horas de teoría	:	2 (Semanal)
2.3 Horas de práctica	:	4 (Semanal)
2.4 Duración del periodo	:	16 semanas
2.5 Condición	:	Obligatorio
2.6 Modalidad	:	■FaceToFace■
2.7 Prerrequisitos	:	CS112. Ciencia de la Computación 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 los cursos introductorios a la informática. En este curso se pretende cubrir los conceptos señalados por la Computing Curricula IEEE(c)-ACM 2001, bajo el enfoque funcional-first. El paradigma orientado a objetos nos permite combatir la complejidad haciendo modelos a partir de abstracciones de los elementos del problema y utilizando técnicas como encapsulamiento, modularidad, polimorfismo y herencia. El dominio de estos temas permitirá que los participantes puedan dar soluciones computacionales a problemas de diseño sencillos del mundo real.

5. OBJETIVOS

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

6. COMPETENCIAS

- 1) S.O. Analizar un problema computacional complejo y aplicar los principios computacionales y otras disciplinas relevantes para identificar soluciones. (**Usar**)
- 3) S.O. Comunicarse efectivamente en diversos contextos profesionales. (**Usar**)
- 5) S.O. Funcionar efectivamente como miembro o líder de un equipo involucrado en actividades apropiadas a la disciplina del programa. (**Usar**)

7. COMPETENCIAS ESPECÍFICAS

Nospecificoutcomes

8. TEMAS

Unidad 1: Introducción a Punteros en C/C++ (5)	
Competencias esperadas: 1	
Temas	Objetivos de Aprendizaje
<ul style="list-style-type: none"> • Declaración de punteros. • Trabajo con punteros: <ul style="list-style-type: none"> Referenciación. Desreferenciación. • Punteros tipados, aritmética de punteros, punteros void. • Punteros a punteros. • Punteros como argumentos de una función-llamada por referencia. 	<ul style="list-style-type: none"> • Introducir en el manejo de punteros, sus operadores y su interacción en la memoria.[Usar] • Demostrar mediante ejemplos los diferentes usos de los operadores con punteros.[Usar] • Demostrar mediante ejemplos el uso de aritmética de punteros. [Usar] • Demostrar mediante ejemplos, las diferentes llamadas a funciones y el uso de punteros. [Usar]
Lecturas : [Nak13], [Str13], [Ree13], [Top13]	

Unidad 2: Programación orientada a objetos (7)	
Competencias esperadas:	
Temas	Objetivos de Aprendizaje
<ul style="list-style-type: none"> ● Diseño orientado a objetos: <ul style="list-style-type: none"> – Descomposición en objetos que almacenan estados y poseen comportamiento – Diseño basado en jerarquía de clases para modelamiento ● Definición de las categorías, campos, métodos y constructores. ● Las subclases, herencia y método de alteración temporal. ● Asignación dinámica: definición de método de llamada. ● Subtipificación: <ul style="list-style-type: none"> – 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. ● Lenguajes orientados a objetos para la encapsulación: <ul style="list-style-type: none"> – privacidad y la visibilidad de miembros de la clase – Interfaces revelan único método de firmas – clases base abstractas ● Uso de colección de clases, iteradores, y otros componentes de la librería estándar. 	<ul style="list-style-type: none"> ● 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 proceduracional/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 [Usar] ● 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) [Usar] ● 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 : [stroustrup13]	

Unidad 3: Manejo de Punteros con arrays (5)	
Competencias esperadas: 1,3	
Temas	Objetivos de Aprendizaje
<ul style="list-style-type: none"> ● Arrays como argumentos de una función. ● Arrays de caracteres y punteros. ● Punteros y Arrays de 2 dimensiones. ● Punteros y arrays multidimensionales. 	<ul style="list-style-type: none"> ● Demostrar el uso de punteros con diferentes tipos de Arrays. [Usar] ● Demostrar la disposición de un array en la memoria y como se manipulan punteros dentro de esos espacios de memoria. [Usar] ● Demostrar el uso de aritmética de punteros y arrays. [Usar]
Lecturas : [Nak13], [Str13], [Ree13], [Top13]	

Unidad 4: Punteros y memoria dinámica (3)	
Competencias esperadas: 1	
Temas	Objetivos de Aprendizaje
<ul style="list-style-type: none"> • Punteros y memoria dinámica - stack vs heap. • Alocación de memoria dinámica en C - malloc, calloc, realloc, free. • 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++. 	<ul style="list-style-type: none"> • Mostrar la estructura de la memoria dentro de un programa y comprender como es que el compilador dispone elementos en el stack y en el heap.[Usar] • Demostrar el uso de las funciones y operadores de asignación de 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 : [Nak13], [Str13], [Ree13], [Top13]	

Unidad 5: Punteros y clases (5)	
Competencias esperadas: 1	
Temas	Objetivos de Aprendizaje
<ul style="list-style-type: none"> • 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. 	<ul style="list-style-type: none"> • 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 estructura de datos, mostrando una implementación simple de listas enlazadas.[Usar]
Lecturas : [Nak13], [Str13], [Ree13], [Top13]	

Unidad 6: Functores (3)	
Competencias esperadas: 1,3	
Temas	Objetivos de Aprendizaje
<ul style="list-style-type: none"> • Definición de functores. • Functores y templates. • Paso de functores a funciones usando parámetros. • Paso de functores a funciones usando templates. • Paso de functores a clases usando parámetros. • Paso de functores a clases usando templates. • Ejemplos y aplicaciones. 	<ul style="list-style-type: none"> • Introducción a los functores. [Usar] • Uso de functores como parámetros a funciones y clases. [Usar] • Uso de functores en funciones y clases a través de templates. [Usar]
Lecturas : [Nak13], [Str13], [Ree13], [Top13]	

9. PLAN DE TRABAJO

9.1 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.

9.2 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.

9.3 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.

10. SISTEMA DE EVALUACIÓN

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

11. BIBLIOGRAFÍA BÁSICA

- [Nak13] S. Nakariakov. *The Boost C++ Libraries: Generic Programming*. CreateSpace Independent Publishing Platform, 2013.
- [Ree13] M. Reese. *Understanding and Using C Pointers*. Inc., 2013.
- [Str13] B Stroustrup. *The C++ Programming Language, 4th edition*. Addison-Wesley, 2013.
- [Top13] H Toppo N. Dewan. *Pointers in C: A Hand on Approach*. Apress, 2013.