Universidad Católica San Pablo (UCSP)
Escuela Profesional de
Ciencia de la Computación
SILABO

CS113. Ciencia de la Computación II (Obligatorio)

2023-I

1. Información general

1.1 Escuela	:	Ciencia de la Computación
1.2 Curso	:	CS113. Ciencia de la Computación II
1.3 Semestre	:	3^{er} Semestre.
1.4 Prerrequisitos	:	CS112. Ciencia de la Computación I. (2^{do} Sem)
1.5 Condición	:	Obligatorio
1.6 Modalidad de aprendizaje	:	Presencial
1.7 horas	:	2 HT; 4 HP;
1.8 Créditos	:	4
1.9 Plan	:	Plan Curricular 2016

2. Profesores

Titular

- Christian Jorge Delgado Polar <cjdelgado@ucsp.edu.pe>
  - Master en Ciencia de la Computación, DCC-UFMG, Brasil, 2007.
- Gustavo Delgado Ugarte <ggdelgado@ucsp.edu.pe>
  - Master en Ingeniería del Software, Escuela Universitaria de Ingeniería Industrial, Informática y Sistemas - UTA, Chile, 2009.

3. Fundamentación del 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 functional-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.

4. Resumen

1. Introducción a Punteros en C/C++ 2. Programación orientada a objetos 3. Manejo de Punteros con arrays 4. Punteros y memoria dinámica 5. Punteros y clases 6. Functores

5. Objetivos Generales

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

6. Contribución a los resultados (*Outcomes*)

Esta disciplina contribuye al logro de los siguientes resultados de la carrera:

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. Contenido

UNIDAD 1: Introducción a Punteros en C/C++ (5)

Competencias: 1

Contenido	**Objetivos Generales**
• Declaración de punteros. • Trabajo con punteros: Referenciación. Desreferenciación. • Punteros tipados, aritmética de punteros, punteros void. • Punteros a punteros. • Punteros como argumentos de una función-llamada por referencia.	• 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: Nakariakov (2013), **stroustrup2013**, **Reese2013**, **Toppo2013**

UNIDAD 2: Programación orientada a objetos (7)

Competencias:

Contenido	**Objetivos Generales**
• Diseño orientado a objetos: – Descomposicion en objetos que almacenan estados y poseen comportamiento – Diseño basado en jerarquia 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: – 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: – privacidad y la visibilidad de miembros de la clase – Interfaces revelan único método de firmas – clases base abstractas • Uso de coleccion de clases, iteradores, y otros componentes de la libreria estandar.	• 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 procedurar/funcional- definiendo una función por cada operación con el cuerdo 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 (codigo compartido y *overriding*) 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, selecionar la forma mas natural por cada lenguaje [Usar]

Lecturas: Stroustrup (2013)

UNIDAD 3: Manejo de Punteros con arrays (5)

Competencias: 1,3

Contenido	**Objetivos Generales**
• 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: Nakariakov (2013), **stroustrup2013**, **Reese2013**, **Toppo2013**

UNIDAD 4: Punteros y memoria dinámica (3)	
Competencias: 1	
Contenido	**Objetivos Generales**
• 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++.	• 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: Nakariakov (2013), **stroustrup2013**, **Reese2013**, **Toppo2013**	

UNIDAD 5: Punteros y clases (5)	
Competencias: 1	
Contenido	**Objetivos Generales**
• 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 estructura de datos, mostrando una implementación simple de listas enlazadas.[Usar]
Lecturas: Nakariakov (2013), **stroustrup2013**, **Reese2013**, **Toppo2013**	

UNIDAD 6: Functores (3)	
Competencias: 1,3	
Contenido	**Objetivos Generales**
• 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.	• 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: Nakariakov (2013), **stroustrup2013**, **Reese2013**, **Toppo2013**	

8. Metodología

1. El profesor del curso presentará clases teóricas de los temas señalados en el programa propiciando la intervención de los alumnos.
2. El profesor del curso presentará demostraciones para fundamentar clases teóricas.
3. El profesor y los alumnos realizarán prácticas
4. Los alumnos deberán asistir a clase habiendo leído lo que el profesor va a presentar. De esta manera se facilitará la comprensión y los estudiantes estarán en mejores condiciones de hacer consultas en clase.

9. Evaluar

Evaluación Continua 1 : 20 %

Examen parcial : 30 %

Evaluación Continua 2 : 20 %

Examen final : 30 %

References

Nakariakov, S. (2013). *The Boost C++ Libraries: Generic Programming.* CreateSpace Independent Publishing Platforml.
Stroustrup, B (2013). *The C++ Programming Language, 4th edition.* Addison-Wesley.