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

CS112. Ciencia de la Computación I (Obligatorio)

2020-I

1. Información general		
1.1 Escuela	:	Ciencia de la Computación
1.2 Curso	:	CS112. Ciencia de la Computación I
1.3 Semestre	:	2^{do} Semestre.
1.4 Prerrequisitos	:	CS111. Programación de Video Juegos. (1^{er} Sem)
1.5 Condición	:	Obligatorio
1.6 Modalidad de aprendizaje	:	Virtual
1.7 horas	:	2 HT; 2 HP; 4 HL;
1.8 Créditos	:	5

2. Profesores

3. Fundamentación del 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.

4. Resumen

1. Visión General de los Lenguajes de Programación 2. Sistemas de tipos básicos 3. Conceptos Fundamentales de Programación 4. Programación orientada a objetos 5. Algoritmos y Diseño 6. Estrategias Algorítmicas 7. Análisis Básico 8. Algoritmos y Estructuras de Datos fundamentales

5. Objetivos Generales

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

6. Contribución a los resultados (*Outcomes*)

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

1) Analizar un problema computacional complejo y aplicar los principios computacionales y otras disciplinas relevantes para identificar soluciones. (**Evaluar**)

2) Diseñar, implementar y evaluar una solución basada en computación para cumplir con un conjunto determinado de requisitos computacionales en el contexto de las disciplinas del programa. (**Evaluar**)

5) Funcionar efectivamente como miembro o líder de un equipo involucrado en actividades apropiadas a la disciplina del programa. (**Familiarizarse**)

6) Aplicar fundamentos de teoria de ciencias de la computación y desarrollo de software para producir soluciones basados en computación. (**Usar**)

7. Contenido

UNIDAD 1: Visión General de los Lenguajes de Programación (1)	
Competencias:	
Contenido	**Objetivos Generales**
• Breve revisión de los paradigmas de programación. • Comparación entre programación funcional y programación imperativa. • Historia de los lenguajes de programación.	• Discutir el contexto histórico de los paradigmas de diversos lenguajes de programación [Familiarizarse]
Lecturas: Stroustrup2013, **Deitel17**	

UNIDAD 2: Sistemas de tipos básicos (2)

Competencias:

Contenido	**Objetivos Generales**

• Tipos como conjunto de valores junto con un conjunto de operaciones.
  – Tipos primitivos (p.e. números, booleanos)
  – Composición de tipos construídos de otros tipos (p.e., registros, uniones, arreglos, listas, funciones, referencias)
• Declaración de modelos (enlace, visibilidad, alcance y tiempo de vida).
• Vista general del chequeo 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 (*generics*), subtipo y sobrecarga [Familiarizarse]
	• Explicar múltiples beneficios y limitaciones de tipificación estática en escritura, mantenimiento y depuración de un software [Familiarizarse]

Lecturas: Stroustrup2013, Deitel17

UNIDAD 3: Conceptos Fundamentales de Programación (6)	
Competencias:	
Contenido	**Objetivos Generales**
• Sintaxis y semántica básica de un lenguaje de alto nivel. • Variables y tipos de datos primitivos (ej., numeros, caracteres, booleanos) • Expresiones y asignaciones. • Operaciones básicas I/O incluyendo archivos I/O. • Estructuras de control condicional e iterativas. • Paso de funciones y parámetros.	• 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] • Escribe un programa que usa E/S de archivos para brindar persistencia a través de ejecuciones múltiples [Usar] • Escoje estructuras de condición y repetición adecuadas para una tarea de programación dada [Evaluar] • Describe el concepto de recursividad y da ejemplos de su uso [Familiarizarse] • Identifica el caso base y el caso general de un problema basado en recursividad [Evaluar]
Lecturas: Stroustrup2013, **Deitel17**	

UNIDAD 4: Programación orientada a objetos (10)

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 • 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 coleccion de clases, iteradores, y otros componentes de la libreria estandar. • 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 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 [Evaluar] • 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) [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, selecionar la forma mas natural por cada lenguaje [Usar]

Lecturas: Stroustrup2013, **Deitel17**

UNIDAD 5: Algoritmos y Diseño (3)

Competencias:

Contenido	**Objetivos Generales**
• Estrategias de solución de problemas – Funciones matemáticas iterativas y recursivas – Recorrido iterativo y recursivo en estructura de datos – Estrategias Divide y Conquistar • Rol de los algoritmos en el proceso de solución de problemas • Estrategias de solución de problemas – Funciones matemáticas iterativas y recursivas – Recorrido iterativo y recursivo en estructura de datos – Estrategias Divide y Conquistar • Conceptos y principios fundamentales de diseño – Abstracción – Descomposición de Program – Encapsulamiento y camuflaje de información – Separación de comportamiento y aplicación	• Discute la importancia de los algoritmos en el proceso de solución de un problema [Familiarizarse] • Discute como un problema puede ser resuelto por múltiples algoritmos, cada uno con propiedades diferentes [Familiarizarse] • Crea algoritmos para resolver problemas simples [Usar] • Usa un lenguaje de programación para implementar, probar, y depurar algoritmos para resolver problemas simples [Usar] • Implementa, prueba, y depura funciones recursivas simples y sus procedimientos [Usar] • Determina si una solución iterativa o recursiva es la más apropiada para un problema [Evaluar] • Implementa un algoritmo de divide y vencerás para resolver un problema [Usar] • Aplica técnicas de descomposición para dividir un programa en partes más pequeñas [Usar] • Identifica los componentes de datos y el comportamiento de mútiples tipos de datos abstractos [Usar] • Implementa un tipo de dato abstracto coherente, con la menor pérdida de acoplamiento entre componentes y comportamientos [Usar] • Identifica las fortalezas y las debilidades relativas entre múltiples diseños e implementaciones de un problema [Evaluar]

Lecturas: Stroustrup2013, **Deitel17**

UNIDAD 6: Estrategias Algorítmicas (3)	
Competencias:	
Contenido	**Objetivos Generales**
• Algoritmos de fuerza bruta. • Algoritmos voraces. • Divide y vencerás. • Bactraking recursivo. • Programación Dinámica.	• Para cada una de las estrategias (fuerza bruta, algoritmo goloso, divide y vencerás, recursividad en reversa y programación dinámica), identifica un ejemplo práctico en el cual se pueda aplicar [Familiarizarse] • Utiliza un enfoque voraz para resolver un problema específico y determina si la regla escogida lo guía a una solución óptima [Evaluar] • Usa un algoritmo de divide-y-vencerás para resolver un determinado problema [Usar] • Usa recursividad en reversa a fin de resover un problema como en el caso de recorrer un laberinto [Usar] • Usa programación dinámica para resolver un problema determinado [Usar] • Determina el enfoque algorítmico adecuado para un problema [Evaluar] • Describe varios métodos basados en heurísticas para resolver problemas [Familiarizarse]
Lecturas: Stroustrup2013, **Deitel17**	

UNIDAD 7: Análisis Básico (2)	
Competencias:	
Contenido	**Objetivos Generales**
• Diferencias entre el mejor, el esperado y el peor caso de un algoritmo.	• Explique a que se refiere con “mejor”, “esperado” y “peor” caso de comportamiento de un algoritmo [Familiarizarse]
Lecturas: Stroustrup2013, **Deitel17**	

UNIDAD 8: Algoritmos y Estructuras de Datos fundamentales (6)	
Competencias:	
Contenido	**Objetivos Generales**
• Algoritmos numéricos simples, tales como el cálculo de la media de una lista de números, encontrar el mínimo y máximo. • Algoritmos de búsqueda secuencial y binaria. • Algoritmos de ordenamiento de peor caso cuadrático (selección, inserción) • Algoritmos de ordenamiento con peor caso o caso promedio en O(N lg N) (Quicksort, Heapsort, Mergesort)	• Implementar algoritmos numéricos básicos [Usar] • Implementar algoritmos de busqueda simple y explicar las diferencias en sus tiempos de complejidad [Evaluar] • Ser capaz de implementar algoritmos de ordenamiento comunes cuádraticos y O(N log N) [Usar] • Discutir el tiempo de ejecución y eficiencia de memoria de los principales algoritmos de ordenamiento, busqueda y hashing [Familiarizarse] • Discutir factores otros que no sean eficiencia computacional que influyan en la elección de algoritmos, tales como tiempo de programación, mantenibilidad, y el uso de patrones específicos de la aplicación en los datos de entrada [Familiarizarse] • Explicar como el balanceamiento del arbol afecta la eficiencia de varias operaciones de un arbol de búsqueda binaria [Familiarizarse] • Demostrar habilidad para evaluar algoritmos, para seleccionar de un rango de posibles opciones, para proveer una justificación por esa selección,y para implementar el algoritmo en un contexto en específico [Evaluar] • Trazar y/o implementar un algoritmo de comparación de string [Usar]
Lecturas: Stroustrup2013, **Deitel17**	

8. Metodología

El profesor del curso presentará clases teóricas de los temas señalados en el programa propiciando la intervención de los alumnos.

El profesor del curso presentará demostraciones para fundamentar clases teóricas.

El profesor y los alumnos realizarán prácticas

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 %