SOCIEDAD PERUANA DE COMPUTACIÓN

Universidad Tecnológica de Panamá (UTP)
Programa Profesional de
Ciencia de la Computación
Sílabo 2023-I

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	:	Híbrido
2.8 Prerrequisitos	:	CS111. Introducción a la Ciencia de la Computación. (1^{er} Sem) CS111. Introducción a la Ciencia de la Computación. (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) S.O. Analizar un problema computacional complejo y aplicar los principios computacionales y otras disciplinas relevantes para identificar soluciones. (**Evaluar**)

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

6) S.O. Aplicar la teoría de la computación y los fundamentos del desarrollo de software para producir soluciones basadas en computación. (**Usar**)

7. TEMAS

Unidad 1: Visión General de los Lenguajes de Programación (1)	
Resultados esperados: 1	
Temas	**Objetivos de Aprendizaje**
• 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: Máquinas virtuales (2)	
Resultados esperados: 1,6	
Temas	**Objetivos de Aprendizaje**
• 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 hadware y software [Familiarizarse] • Diferenciar emulacion y el aislamiento [Familiarizarse] • Evaluar virtualización de compensaciones [Evaluar]
Lecturas : [**Stroustrup2013**], [**Deitel17**]	

Unidad 3: Sistemas de tipos básicos (6)	
Resultados esperados: 1,6	
Temas	**Objetivos de Aprendizaje**
• 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 4: Conceptos Fundamentales de Programación (10)	
Resultados esperados: 1,6	
Temas	**Objetivos de Aprendizaje**
• Diseño orientado a objetos: – Descomposicion en objetos que almacenan estados y poseen comportamiento – Diseño basado en jerarquia de clases para modelamiento • Tanto para tipo primitivo y un tipo compuesto, describir de manera informal los valores que tiene dicho tipo [Familiarizarse] • Variables y tipos de datos primitivos (ej., numeros, caracteres, booleanos) • Expresiones y asignaciones. • Estructuras de control condicional e iterativas.	• 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] • Escoje estructuras de condición y repetición adecuadas para una tarea de programación dada [Evaluar]
Lecturas : [**Stroustrup2013**], [**Deitel17**]	

Unidad 5: Funciones (3)	
Resultados esperados: 1,6	
Temas	**Objetivos de Aprendizaje**
• Paso de funciones y parámetros. • Paso de parámetros • Sobrecarga en funciones • Fundamentos de la recursidad • Conceptos de plantillas en 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 concpeto de plantillas asociado a la ncesidad de crear funciones genéricas.[Usar]
Lecturas : [**Stroustrup2013**], [**Deitel17**]	

Unidad 6: Arreglos y Punteros (3)	
Resultados esperados: 1,6	
Temas	**Objetivos de Aprendizaje**
• Definición de arreglos • Arreglos multidimensionales • Fundamentos sobre punteros • Administración dinámica de memoria • Conceptos avanzados de Punteros	• 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 evalua 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 evalua el concepto de puntero a puntero, puntero a función, entre otros conceptos.[Evaluar]
Lecturas : [**Stroustrup2013**], [**Deitel17**]	

Unidad 7: Programación orientada a objetos (2)	
Resultados esperados: 1,6	
Temas	**Objetivos de Aprendizaje**
• 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 8: Plantillas y STL (2)	
Resultados esperados: 1,6	
Temas	**Objetivos de Aprendizaje**
• Definición de plantillas en clases • Conceptos básicos sobre la Standard Template Library (STL)	• 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 : [**Stroustrup2013**], [**Deitel2017**]	

Unidad 9: Conceptos Avanzados (2)	
Resultados esperados: 1,6	
Temas	**Objetivos de Aprendizaje**
• Definición de sobrecarga de operadores • Manipulación de entrada y salida de datos (I/O) • Patrones de diseño	• Entiende los conceptos de sobrecarga de operadores. [Familiarizarse] • Implementa la sobrecarga de operadores permitidos en el lenguaje de programación. [Usar] • Entiende los conceptos de manipulación de archivos. [Familiarizarse] • Crea programas de lectura y escrita en archivos. [Usar] • Entiende los conceptos de patrones de diseño. [Familiarizarse]
Lecturas : [**Stroustrup2013**], [**Deitel2017**]	

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