CIENCIA DE LA COMPUTACIÓN

Programación Orientada a Objetos II

4 CRÉDITOS

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

CE2A

ÍNDICE

UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA
SILABO 2020-2

1. ASIGNATURA

CS1103 - Programación Orientada a Objetos II

2. DATOS GENERALES

2.1 Créditos: cuatro (4) créditos
2.2 Horas de teoría: dos (2) semanales
2.3 Horas de práctica: tres (3) semanales
2.4 Duración del período: dieciséis (16) semanas
2.5 Condición:
- Obligatorio para Ciencia de la Computación

2.6 Modalidad: Virtual
2.7 Requisitos:
- CS1102 - Programación Orientada a Objetos 1

3. PROFESORES

3.1 Profesor coordinador del curso

Rubén Demetrio Rivas Medina (rrivas@utec.edu.pe)
Horario de atención: miércoles 9:00 a 10:00 am

3.2 Profesor(es) instructor(es) del curso

Rubén Demetrio Rivas Medina (rrivas@utec.edu.pe)
Horario de atención: miércoles 9:00 a 10:00 am

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 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 permitiría que los participantes puedan dar soluciones computacionales a problemas de diseño sencillos del mundo real.

5. OBJETIVOS

Sesión 1: Definir conceptos fundamentales y su relación con la programación orientada a objetos y la librería estándar C++.
Sesión 2: Definir estructuras genéricas utilizando templates, desarrollar listas consecutivas y listas enlazadas genéricas.
Sesión 3: Explicar y clasificar los algoritmos y estrategias de solución de algoritmos, explicación de los patrones y la relación y diferencia entre ambos.
Sesión 4: Analizar y detallar métodos que permitan evaluar la eficiencia de ejecución de un algoritmo.
Sesión 5: Evaluación del aprendizaje parcial con una práctica calificada.
Sesión 6: Definir y detallar algoritmos numéricos, de búsqueda y ordenamiento.
Sesión 7: Definir y detallar estructuras básicas: pilas y stacks.
Sesión 8: Definir y detallar heap y hash table y sus usos.
Sesión 9: Definir y detallar árboles de búsqueda binaria y algoritmos asociados y las formas de implementarlo.
Sesión 10: Definir y detallar grafos y algoritmos asociados y las formas de implementarlo.
Sesión 11: Evaluación del aprendizaje parcial con una práctica calificada.
Sesión 12: Explicar y detallar la programación concurrente, propiedades y herramientas brindadas por C++ estándar.
Sesión 13: Explicar y detallar la programación reactiva, aplicar algunas herramientas en interfaces gráficas.
Sesión 14: Definir y detallar las fases de desarrollo de software y su relación con la ingeniería de requerimientos.
Sesión 15: Evaluación del aprendizaje parcial con una práctica calificada.
Sesión 16: Evaluación del aprendizaje parcial con la exposición y presentación del proyecto.

6. COMPETENCIAS

A. Aplicar conocimientos de computación y de matemáticas apropiadas para la disciplina. (Nivel 2)
   El estudiante desarrolla programas utilizando conceptos de diseño y programación orientados a objetos, estructuras fundamentales y algoritmos, considerando los criterios apropiados de algoritmo y diseño.

B. Analizar problemas e identificar y definir los requerimientos computacionales apropiados para su solución. (Nivel 2)
   El estudiante analiza un problema, identifica los requerimientos y diseña un programa que cumpla con los requerimientos.

C. Trabajar efectivamente en equipos para cumplir con un objetivo común. (Nivel 2)
   El estudiante podrá asumir alguno de los roles que pueda requerir el desarrollo de un proyecto.

7. RESULTADOS DE APRENDIZAJE

Al final del curso de Programación Orientada a Objetos 2 se espera:

RA 1. Que el estudiante sea capaz de modelar programas orientados a objetos utilizando principios de abstracción, generalización, eficiencia y reutilización de código.

RA 2. Que el estudiante sea capaz de seleccionar, para el diseño de programas, las estructuras de datos y algoritmos fundamentales basados en criterios de eficiencia y reutilización.

RA 3. Que el estudiante sea capaz de implementar tipos de datos abstractos y estructuras de datos fundamentales, como queue, stacks, bags, hash, heaps, árboles y grafos.

RA 4. Que el estudiante sea capaz de usar un paradigma de diseño para el desarrollo de un sistema de software básico y explicar cómo los principios de diseño del sistema se han aplicado en este desarrollo.

8. TEMAS

1. **Conceptos Fundamentales de Programación y Programación orientada a objetos.**
   - 1.1. Sistema de tipos de datos
   - 1.2. Estructuras básicas de control y de datos
   - 1.3. Funciones, Procedimientos, operadores y subrutinas
   - 1.4. Tipos estáticos vs tipos Dinámicos, beneficios de su complementación
   - 1.5. Elementos de un lenguaje de programación orientado a objetos campos, métodos y constructores, destructores
   - 1.6. Características de la Programación Orientada a Objetos: Encapsulamiento, Sub-tipificación, Agregación y Composición, Abstracción, Polimorfismo
   - 1.7. Paradigmas complementarios: Funcional, Procedural y Declarativos.
   - 1.8. Colecciones: secuenciales y consecutivas; iteradores y componentes de la librería estándar
   - 1.9. Tipos genéricos
2. **Algoritmos y Diseño**
   - 2.1. Conceptos y propiedades de los algoritmos
   - 2.2. Rol de los algoritmos en el proceso de solución de problemas
   - 2.3. Estrategias de solución de problemas
   - 2.4. Conceptos y principios fundamentales de diseño
   - 2.5. Patrones y Principios del diseño orientado a objetos
3. **Análisis Básico de algoritmos**
   - 3.1. Diferencias entre el mejor, el esperado y el peor caso de un algoritmo.
   - 3.2. Análisis asintótico de complejidad de cotas superior y esperada.

3.3. Definición formal de la Notación Big O.
3.4. Clases de complejidad como constante, logarítmica, lineal, cuadrática y exponencial.
3.5. Medidas empíricas de desempeño.
3.6. Compensación entre espacio y tiempo en los algoritmos.
3.7. Uso de la notación Big O.
3.8. Notación Little o, Big omega y Big theta.
3.9. Análisis de algoritmos iterativos y recursivos.
3.10. Teorema Maestro y Árboles Recursivos.

4. Algoritmos y Estructuras de Datos fundamentales

4.1. Algoritmos numéricos.
4.2. Algoritmos de búsqueda secuencial y binaria.
4.3. Algoritmos de ordenamiento.
4.4. Estructuras Simples: Pilas y Colas
4.5. Heap y Hash Tables
4.6. Algoritmos de cadenas/texto
4.7. Árboles de búsqueda binaria
4.8. Grafos y algoritmos de recorrido en grafos

5. Programación Concurrente

5.1. Concurrencia y propiedades de los sistemas concurrentes
5.2. Modelos de Programación Concurrente y arquitecturas de computación paralela
5.3. Manejo de Hilos, memoria compartida, race condition, mutex y locks
5.4. Sincronización de data y mensajería entre hilos: promise/future

6. Programación reactiva y dirigida por eventos

6.1. Eventos y controladores de eventos, Usos canónicos.
6.2. Uso de framework reactivos (Patrón Observador).
6.3. Patrón MVC: modelo, vista y controlador

7. Diseño de Software e Ingeniería de Requerimientos

7.1. Fases del desarrollo de software y el papel de la ingeniería de requerimientos y el diseño de software
7.2. Relaciones entre los requisitos y diseños: La transformación de modelos, el diseño de los contratos, invariantes.
7.3. Diseño de software, niveles de abstracción arquitectónico y detallado, separación de intereses, ocultamiento de información, de acoplamiento y de cohesión, de reutilización de estructuras estándar.

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.
A lo largo del curso se proporcionan diferentes lecturas, las cuales podrían ser evaluadas. El uso de herramientas Online permite a cada estudiante acceder a la información del curso, e interactuar fuera de aula con el profesor y con los otros estudiantes.

9.2 Sesiones de teoría

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 de práctica (laboratorio o taller)

Semanalmente en la sesión de practica se desarrollará junto con los alumnos retos de programación utilizando una metodología activa, promoviendo el trabajo individual en búsqueda de desarrollar sus habilidades programación, análisis y diseño y el trabajo grupal buscando que asuman un rol en un equipo de trabajo, fomentando la integración de los componentes de software.

10. SISTEMA DE EVALUACIÓN

10.1 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.

10.2 Sesiones de Laboratorio:

Para verificar que los alumnos hayan alcanzado el logro planteado para cada una de las unidades de aprendizaje, realizarán actividades que les permita aplicar los conocimientos adquiridos durante las sesiones de teoría y se les propondrá retos que permitan evaluar el desempeño de los alumnos.

10.3 Exposiciones individuales o grupales:

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.

10.4 Lecturas:

A lo largo del curso se proporcionan diferentes lecturas, las cuales podrían ser evaluadas. El uso de herramientas Online permite a cada estudiante acceder a la información del curso, e interactuar fuera de aula con el profesor y con los otros estudiantes.

	TEORÍA	**PRÁCTICA Y/O LABORATORIO**
EVALUACIÓN	0.25 ∗ Proyecto (P) 0.10 ∗ Evaluación Continua (C1) 0.10 ∗ Evaluación Continua (C2)	0.20 ∗ Práctica Calificada (PC1) 0.20 ∗ Práctica Calificada (PC2) 0.15 ∗ Práctica Calificada (PC3)
	45%	55%
	100%	

La ponderación de la evaluación se hará si ambas partes están aprobadas o siguiendo los parámetros decididos por la dirección de la carrera. Indique este aspecto a continuación del asterisco.

Las rúbricas que permitirán medir las actividades más significativas del curso y que, además se relacionan con la evaluación de las competencias del estudiante son: Proyecto 1 : enlace

11. REFERENCIAS BIBLIOGRÁFICAS

Lippman, S., Lajoie J., Moo B. E. (2012). C++ Primer (5th Edition). Addison-Wesley Professional.

Stroustrup, B. (2013). The C++ Programming Language, Fourth edition. Pearson Education Inc.

Vandervoorde, D., Nicolai, J. y Douglas G. (2017). C++ Templates:The Complete Guide (2nd edition). Addison-Wesley.

Williams, A. (2019). C++ Concurrency in Action. (2nd edition). Manning Publications Co.

Pai, P. y Abraham, P. (2018). C++ Reactive Programming. Packt.

Sedgewick, R. y Wayne K. (2011). Algorithms, Fourth edition. Addison-Wesley.

Martin, R. (2014), Agile Software Development, Principles, Patterns, and Practices, Pearson Education Limited.

Booch, G., Maksimchuk R., Engle, M. W., Young, B. J., Conallen, J. y Houston, K. A. (2007), Object-Oriented Analysis and Design with Applications, Third Edition. Addison-Wesley.