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 2021-1

1. ASIGNATURA

CS1103 - Programación Orientada a Objetos II

2. DATOS GENERALES

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

2.7 Modalidad: Virtual
2.8 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, desarrollar listas consecutivas.

Sesión 2: Programación genéricas utilizando templates, desarrollar listas enlazadas genéricas.

Sesión 3: Analizar y aplicar la librería estándar C++: arquitectura de la librería, contenedores.

Sesión 4: Analizar y aplicar la librería estándar C++: iteradores, algoritmos y callable.

Sesión 5: Analizar y aplicar métodos que permitan evaluar la eficiencia temporal y espacial de ejecución de un algoritmo.

Sesión 6: Evaluar el aprendizaje parcial con una práctica calificada.

Sesión 7: Definir estructuras básicas: pilas y stacks y aplica en solución de problemas.

Sesión 8: Definir priority queue, heap y hash table y aplica en solución de problemas.

Sesión 9: Definir métodos de ordenamiento: merge, quick y heap.

Sesión 10: Definir árboles binarios y aplicarlo en el algoritmo de búsqueda binaria.

Sesión 11: Definir y detallar grafos y algoritmos asociados y las formas de implementarlo.

Sesión 12: Evaluación del aprendizaje parcial con una práctica calificada.

Sesión 13: Explicar y detallar la programación concurrente, propiedades y herramientas brindadas por C++ estándar.

Sesión 14: Explicar y detallar los patrones de diseño, categorías, ver patrones de creación, estructurales y de comportamiento, aplicarlo en el desarrollo de un proyecto.

Sesión 15: Explicar patrones funcionales y concurrentes, aplicarlo en el desarrollo de un proyecto.

Sesión 16: Evaluar el aprendizaje parcial con una práctica calificada, exposición y presentación del proyecto final.

6. COMPETENCIAS Y CRITERIOS DE DESEMPEÑO

Los criterios de desempeño que se van a trabajar en este curso son:

1.3. Aplica conocimientos de computación apropiados para la solución de problemas definidos y sus requerimientos en la disciplina del programa. (Nivel 2)

2.3. Investiga literatura de computación y otras disciplinas relevantes en el dominio. (Nivel 2)

3.1. Diseña y evalúa sistemas, componentes o procesos que satisfacen las necesidades específicas. (Nivel 2)

4.1. Crea, selecciona, adapta y aplica técnicas, recursos y herramientas modernas para la práctica de la computación y comprende sus limitaciones. (Nivel 2)

7. RESULTADOS DE APRENDIZAJE

Al final del curso de Programación Orientada a Objetos 2 se espera que el estudiante sea capaz de:

RA1. Implementar programas utilizando los paradigmas de programación orientada a objetos, programación genérica y programación concurrente, además del uso de estructuras de datos fundamentales.

RA2. Identificar tendencias tecnológicas y fuentes de documentación útiles para soluciones computacionales.

RA3. Descomponer el proceso de desarrollo de software utilizando metodologías de análisis y diseño orientado a objetos.

RA4. Utilizar IDEs de desarrollo, modelamiento de diagramas orientados a objetos, herramientas de test y pruebas unitarias como herramientas de control de versiones y documentación.

8. TEMAS

1. **Conceptos fundamentales de la programación y programación orientada a objetos.**
   - 1.1. Sistema de tipos de datos. Clasificación y beneficio de uso.
   - 1.2. Sobrecarga de operadores y funciones, namespace y librerías
   - 1.3. Elementos de un lenguaje de programación orientado a objetos campos, métodos y constructores, destructores, funciones friend.
   - 1.4. Características de la programación orientada a objetos: encapsulamiento, sub-tipificación, agregación y composición, abstracción, polimorfismo. Proceso de especialización y generalización.
   - 1.5. Paradigmas complementarios: funcional, procedural y declarativos.

2. Programación genérica

2.1. Templates de clases, template de funciones, variadic templates, especializaciones y características generales.
2.2. Tipos de callable: lambda, std::function, puntero a funciones y functores.
2.3. Arquitectura de la librería estándar, contenedores, iteradores algoritmos y callable.

3. Introducción al análisis de algoritmos

3.1. Medidas empíricas de desempeño.
3.2. Función de tiempo basada en el input del recurso (tiempo, espacio)
3.3. Cota superior asintótica, notación Big O, definición formal.
3.4. Diferencias entre el mejor, el esperado y el peor caso de un algoritmo.
3.5. Análisis práctico de las funciones típicas de orden de crecimiento: constante, logarítmica, lineal, cuadrática, polinómicas y exponenciales.
3.6. Análisis práctico de algoritmos iterativos y recursivos.
3.7. Aplicación del teorema maestro en el análisis de complejidad.

4. Algoritmos y estructuras de datos fundamentales

4.1. Pilas y colas.
4.2. Priority queues, heap y hash tables.
4.3. Algoritmos de ordenamiento: merge, quicksort y heap.
4.4. Árboles de búsqueda binaria
4.5. Algoritmos de búsqueda: secuencial y binaria usando árbol binario.
4.6. 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: memoria compartida, mensajería y paralelismo.
5.3. Hilos, creación de hilos en varios escenarios memoria compartida, race condition, data race, uso de mutex y lock_guards.
5.4. Sincronización de data y mensajería entre hilos: promise/future

6. Patrones de diseño

6.1. Descripción e importancia de su uso.
6.2. Categorías de patrones: creación, estructura, comportamiento, funcionales y concurrentes.
6.3. Ejemplos de patrones por categorías.

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 práctica 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.

EVALUACIÓN	TEORÍA	PRÁCTICA Y/O LABORATORIO

	Proyecto **P1 (**25%) Evaluación Continua **C1** (10%) Evaluación Continua **C2** (10%)	Práctica Calificada **PC1** (20%) Práctica Calificada **PC2** (20%) Práctica Calificada **PC3** (15%)
	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, **PC1:** 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.
- Dmitri Nesteruk. (2018). Design Patterns in Modern C++: Reusable Approaches for Object-Oriented Software Design. APress.
- Sedgewick, R. y Wayne K. (2011). Algorithms, Fourth edition. Addison-Wesley.