Sílabo

Malla 2021

DEPARTAMENTO

Departamento de Computer Science

CURSO

Programación III

MALLA

2021

MODALIDAD

PRESENCIAL

CREDITOS

4

REGLAS INTEGRIDAD ACADÉMICA

Todo estudiante matriculado en una asignatura de la Universidad de Ingeniería y Tecnología tiene la obligación de conocer y cumplir las reglas de integridad académica, cuya lista a continuación es de carácter enunciativo y no limitativo, ya que el/la docente podrá dar mayores indicaciones:

1. La copia y el plagio son dos infracciones de magnitud muy grave en la Universidad de Ingeniería y Tecnología (UTEC) conforme a lo establecido en el Reglamento de Disciplina de los Estudiantes. Tienen una sanción desde 2 semestres de suspensión hasta la expulsión.
2. Si se identifica la copia o plagio en evaluaciones individuales, el/la docente puede proceder a anular la evaluación.
3. Si la evaluación es personal o grupal-individual, la interacción entre equipos o compañeros se considera copia o plagio, según corresponda. Si la evaluación calificada no indica que es grupal, se presume que es individual.
4. La copia, plagio, el engaño y cualquier forma de colaboración no autorizada no serán tolerados y serán tratados de acuerdo con las políticas y reglamentos de la UTEC, implicando consecuencias académicas y sanciones disciplinarias.
5. Aunque se alienta a los estudiantes a discutir las tareas y trabajar juntos para desarrollar una comprensión más profunda de los temas presentados en este curso, no se permite la presentación del trabajo o las ideas de otros como propios. No se permite el plagio de archivos informáticos, códigos, documentos o dibujos.
6. Si el trabajo de dos o más estudiantes es sospechosamente similar, se puede aplicar una sanción académica a todos los estudiantes, sin importar si es el estudiante que proveyó la información o es quien recibió la ayuda indebida. En ese sentido, se recomienda no proveer el desarrollo de sus evaluaciones a otros compañeros ni por motivos de orientación, dado que ello será considerado participación en copia.
7. El uso de teléfonos celulares, aplicaciones que permitan la comunicación o cualquier otro tipo de medios de interacción entre estudiantes está prohibido durante las evaluaciones o exámenes, salvo que el/la docente indique lo contrario de manera expresa. Es irrelevante la razón del uso del dispositivo.
8. En caso exista algún problema de internet durante la evaluación, comunicarse con el/la docente utilizando el protocolo establecido. No comunicarse con los compañeros dado que eso generará una presunción de copia.
9. Se prohíbe tomar prestadas calculadoras o cualquier tipo de material de otro estudiante durante una evaluación, salvo que el/la docente indique lo contrario.
10. Si el/la docente encuentra indicios de obtención indebida de información, lo que también implica no cumplir con las reglas de la evaluación, tiene la potestad de anular la prueba, advertir al estudiante y citarlo con su Director de Carrera. Si el estudiante no asiste a la citación, podrá ser reportado para proceder con el respectivo procedimiento disciplinario. Una segunda advertencia será reportada para el inicio del procedimiento disciplinario correspondiente.
11. Se recomienda al estudiante estar atento/a a los datos de su evaluación. La consignación de datos que no correspondan a su evaluación será considerado indicio concluyente de copia.

UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

SÍLABO DEL CURSO

1. ASIGNATURA

CS2013 – Programación III

2. DATOS GENERALES

2.1 Ciclo: NIVEL 3
2.2 Créditos: 4
2.3 Condición: Obligatorio para Ciencia de la Computación
2.4 Idioma de dictado: Español
2.5 Requisitos: CS1112 - Programación II

3. INTRODUCCIÓN AL CURSO

Este curso, de naturaleza teórica y práctica, proporcionará a los participantes una comprensión profunda de los principios y prácticas clave en programación orientada a objetos, programación genérica y programación concurrente en C++. Desde la definición de conceptos fundamentales hasta la implementación de estructuras de datos esenciales, cada sesión se enfoca en habilidades específicas.. Este curso prepara a los estudiantes para abordar problemas computacionales complejos con un enfoque integrado y una sólida base en las prácticas contemporáneas de desarrollo de software.

4. OBJETIVOS

- Sesión 1: Definir conceptos fundamentales y su relación con la programación orientada a objetos.
- Sesión 2: Utilizar templates en soluciones computacionales genéricas.
- Sesión 3: Usar la programación genérica, arquitecturas de la librería estándar: contenedores e iteradores.
- Sesión 4: Usar la programación genérica, arquitecturas de la librería estándar : tipos de Callables.
- Sesión 5: Analizar la complejidad algorítmica y determinar la eficiencia de un algoritmo en espacio y tiempo.
- Sesión 6: Aplicar patrones de diseño de creación, estructurales y comportamiento en el desarrollo de un proyecto.
- Sesión 7: Usar la programación concurrente, propiedades y herramientas brindadas por C++.
- Sesión 8: Implementar estructuras de datos básicas como pilas y colas.
- Sesión 9: Implementar estructuras de datos como heap, priority queue, ordenamiento heap.

- Sesión 10: Implementar estructuras de datos como Hash table, árboles binarios de búsqueda.
- Sesión 11: Implementar grafos, y los algoritmos de recorrido (BFS y DFS).
- Sesión 12: Implementar algoritmos para búsqueda de Árboles Expandidos Mínimos (Prim y Kruskal) y Rutas óptimas (Dijkstra, Floyd).

5. COMPETENCIAS Y CRITERIOS DE DESEMPEÑO

Competencias Especificas ABET - COMPUTACION

- Analizar un problema computacional complejo y aplicar principios de computación y otras disciplinas relevantes para identificar soluciones.
- Diseñar, implementar y evaluar una solución computacional para satisfacer un conjunto determinado de requerimientos computacionales en el contexto de la disciplina del programa.
- Aplicar la teoría de la ciencia de la computación y los fundamentos de desarrollo de software para producir soluciones basadas en computación. [CS]

6. RESULTADOS DE APRENDIZAJE

- Descomponer el proceso de desarrollo de software utilizando metodologías de análisis y diseño orientado a objetos.
- Utilizar IDEs de desarrollo, modelamiento de diagramas orientados a objetos, herramientas de test y pruebas unitarias como herramientas de control de versiones y documentación.
- Identificar tendencias tecnológicas y fuentes de documentación útiles para soluciones computacionales.
- Implementar programas utilizando los paradigmas de programación orientada a objetos, programación genérica y programación concurrente, además del uso de estructuras de datos fundamentales.

7. TEMAS

1. Conceptos fundamentales de la programación orientada a objetos

1.1. Clases
1.2. Polimorfismo
1.3. Sobrecarga

2. Programación genérica

2.1. Template de clases, funciones: Especialización y Generación
2.2. Variadic templates
2.3. Callables
2.4. Arquitectura y detalles de la librería estándar.

3. Introducción al análisis de algoritmos

1.1. Medición empírica del desempeño
1.2. Función de tiempo
1.3. Orden de crecimiento asintótico: orden de crecimiento y notación Big O

4. Programación concurrente

4.1. Modelos de programación concurrente: memoria compartida, mensajería y paralelismo
4.2. Hilos, race condition, sincronización de hilos, uso de mutex y lock_guards
4.3. Mensajería entre hilos: promise/future, async

5. Patrones de diseño

5.1. Descripción e importancia de su uso
5.2. Categorías de patrones: creación, estructura, comportamiento
5.3. Ejemplos de patrones por categorías

6. Estructuras fundamentales

6.1. Stack y Queue
6.2. Heap, priority queue y árboles
6.3. Tabla Hash, árboles binarios de búsqueda

7. Grafos y algoritmos esenciales

7.1. Tipos de grafos
7.2. Recorrido de grafos BFS, DFS
7.3. Arboles Expandidos
7.4. Algoritmos de búsqueda de árboles expandidos mínimos: Prim, Kruskal
7.5. Algoritmos de rutas óptimas: Dijkstra, Floyd

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.
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 del aula con el profesor y con los otros estudiantes.

8.2 Sesiones de teoría

Las sesiones de teoría se llevan a cabo en clases magistrales donde se realizan actividades que propicien un aprendizaje activo, con dinámicas que permitan a los estudiantes interiorizar los conceptos.

8.3 Sesiones de práctica (laboratorio o taller)

Semanalmente en la sesión de práctica se desarrollará junto con los alumnosretos de programación utilizando una metodología activa, promoviendo el trabajoindividual en búsqueda de desarrollar sus habilidades Programación, análisis ydiseño y el trabajo grupal buscando que asuman un rol en un equipo de trabajo,fomentando la integración de los componentes de software.

9. SISTEMA DE EVALUACIÓN

El curso consta de los siguientes espacios de evaluación:

	Teoría
Evaluación	TEORÍA 50% Proyecto P1 (20%) Evaluación Continua C1 (15%) Evaluación Continua C2 (15%) **LABORATORIO 50%** Práctica Calificada PC1 (15%) Práctica Calificada PC2 (20%) Práctica Calificada PC3 (15%) **Nota** :La ponderación de la evaluación se hará si ambas partes están aprobadas.
	100%

10. REFERENCIAS BIBLIOGRÁFICAS

[1] S. L. J. M. B. E. Lippman, C++ Primer (5th Edition), Massachusetts:Addison-Wesley, 2012.

[2] B. Stroustrup, The C++ Programming Language (4th Edition), Michigan:Addison-Wesley, 2013.
[3] D. N. J. y. D. G. Vandevoorde, C++ Templates: The Complete Guide (2ndedition), Addison-Wesley, 2017.
[4] A. Williams, C++ Concurrency in Action. (2nd edition), New York: ManningPublications Co., 2019.
[5] D. Nesteruk, Design Patterns in Modern C++: Reusable Approaches forObject-Oriented Software Design, New York: APress, 2018.
[6] R. y. W. K. Sedgewick, Algorithms, (4th edition), Massachusetts: Addison-Wesley,2011.
[7] T. L. C. R. R. S. C. Cormen, Introduction to Algorithms, fourth edition,Massachusetts: The MIT Press, 2022.