Sílabo

Malla 2021

DEPARTAMENTO

Departamento de Computer Science

CURSO

Programación II

MALLA

2021

MODALIDAD

Blended

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

CS1112 – Programación II

2. DATOS GENERALES

2.1 Ciclo: NIVEL 2,NIVEL 5,NIVEL 3
2.2 Créditos: 4
2.3 Condición: Obligatorio
2.4 Idioma de dictado: Español
2.5 Requisitos: CS1111 - Programación I

3. INTRODUCCIÓN AL CURSO

Este curso de Programación II, de naturaleza teórica y práctica, proporciona una sólida base en C++, abordando desde los fundamentos hasta conceptos avanzados de la Programación Orientada a Objetos. La programación orientada a objetos enfoca el diseño de la solución al objeto, que es el ente manejado por el algoritmo de solución al problema enfocado. Este paradigma brinda una serie de ventajas de optimización y estabilidad del código, lo que permite al estudiante, obtener un producto de mayor calidad y valor en el mercado computacional.

4. OBJETIVOS

- Sesión 1 Comprender el diseño de un algoritmo y la estructura de un programa en C++, conocer y usar tipos de variables y operadores, conocer y usar estructuras de control.
- Sesión 2 Conocer y usar estructuras de control.
- Sesión 3 Entender el concepto de variables globales. Comprender el concepto de transferencia por valor y por referencia. Usar funciones recursivas y compararlas con no-recursivas.
- Sesión 4 Comprender la definición y uso de punteros.
- Sesión 5 Comprender y usar la memoria de forma optimizada con punteros.
- Sesión 6 Utilizar arreglos dinámicos en un código en C++.
- Sesión 7 Usar vectores en sus variaciones para optimizar el desempeño de un código en C++.
- Sesión 8 Comprender el proceso de abstracción y encapsulamiento y la relación clase-objeto. Utilizar constructores/destructores de objetos de clase. Optimizar el código con objetos y arreglos dinámicos.
- Sesión 9 Optimizar el código con objetos y arreglos dinámicos.
- Sesión 10 Comprender y utilizar los conceptos de asociación, composición y agregación.

- Sesión 11 Usar herencia y funciones virtuales en el diseño de un algoritmo en C++.
- Sesión 12 Utilizar polimorfismo en casos de herencia para mejorar organización y legibilidad de código.
- Sesión 13 Comprender las ventajas de utilización de sobrecarga de operadores. Aplicar este concepto en casos diversos de programación.
- Sesión 14 Usar plantillas para reutilizar un código fuente de programación. Conocer y utilizar archivos para optimizar el manejo de data de entrada y salida en un código de programación.

5. COMPETENCIAS Y CRITERIOS DE DESEMPEÑO

Competencias Especificas - NEGOCIOS

- Estructura, analiza y relaciona situaciones de negocio, tomando decisiones acertadas a partir del entendimiento del consumidor, los mercados, los procesos, las organizaciones y las finanzas.

Competencias Especificas ABET - INGENIERIA

- La capacidad de identificar, formular y resolver problemas complejos de ingeniería mediante la aplicación de principios de ingeniería, ciencias y matemáticas.

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.

6. RESULTADOS DE APRENDIZAJE

- Implementar soluciones computacionales utilizando el paradigma de la Programación Orientada a Objetos, técnicas de manejo de clases y objetos, estructuras de control y funciones para resolver problemas de Computación e Ingeniería
- Usar los pilares de la programación orientada a objetos: herencia, polimorfismo, asociación, agregación y composición; en la implementación de soluciones computacionales.
- Desarrollar códigos de programación a través del diseño de soluciones a problemas que requieren un planteamiento orientado a objetos.
- Implementar soluciones computacionales utilizando el paradigma de la Programación Orientada a Objetos, técnicas de manejo de clases y objetos, estructuras de control y funciones para resolver problemas de Computación e Ingeniería
- Usar los pilares de la programación orientada a objetos: herencia, polimorfismo, asociación, agregación y composición; en la implementación de soluciones computacionales.
- Desarrollar códigos de programación a través del diseño de soluciones a problemas que requieren un planteamiento orientado a objetos.

- Implementar soluciones computacionales utilizando el paradigma de la Programación Orientada a Objetos, técnicas de manejo de clases y objetos, estructuras de control y funciones para resolver problemas de Computación e Ingeniería
- Usar los pilares de la programación orientada a objetos: herencia, polimorfismo, asociación, agregación y composición; en la implementación de soluciones computacionales.
- Desarrollar códigos de programación a través del diseño de soluciones a problemas que requieren un planteamiento orientado a objetos.

7. TEMAS

1. Introducción al diseño de algoritmos y programación en C++

1.1. Diseño de un algoritmo y estructura de un programa en C++ (rol de los algoritmos en el proceso de solución de problemas)
1.2. Variables y tipos de datos primitivos (ej., números, caracteres, booleanos) . Tipos de datos y operadores
1.3. Estructuras de control selectivas y repetitivas

2. Funciones y recursividad

2.1. Scope o ámbito: variables globales
2.2. Paso de funciones y parametros. Transferencia por valor y por referencia
2.3. Funciones matematicas iterativas y recursivas

3. Punteros

3.1. Definición y manejo de punteros
3.2. Análisis de punteros en código
3.3. Manejo de memoria (dinámica)
3.4. Arreglos dinámicos (arrays, vectores, matrices)

4. Vectores

4.1. Definición y construcción
4.2. Operadores básicos

5. Programación Orientada a Objetos

5.1. Proceso de abstracción
5.2. Relación clase-objeto
5.3. Métodos de acceso y modificación (getters y setters)
5.4. Constructores y destructores
5.5. Objetos dinámicos
5.6. Arreglos de objetos

6. Relación entre clases

6.1. Asociación

6.2. Composición
6.3. Agregación
6.4. Herencia (funciones virtuales)

7. Polimorfismo

7.1. Polimorfismo para-métrico o genérico
7.2. Polimorfismo de inclusión o herencia
7.3. Herencia de tipos con Override

8. Sobrecarga de operadores

8.1. Funciones amigas
8.2. Sobrecarga de operadores
8.3. Casos particulares de sobrecarga

9. Templates y archivos

9.1. Definición de plantillas (templates)
9.2. Archivos. Definición y modos de uso

8. PLAN DE TRABAJO

8.1 Metodología

El curso presenta una metodología de aprendizaje activo, basada en la resolución de problemas de una manera práctica con enfoque al conocimiento adquirido en sesiones teóricas.
Adicionalmente, existe aprendizaje a través de la ejecución de un proyecto, de esta manera, se anima y motiva al estudiante a que continúe con su proceso de aprendizaje.

8.2 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.

8.3 Sesiones de práctica (laboratorio o taller)

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

9. SISTEMA DE EVALUACIÓN

El curso consta de los siguientes espacios de evaluación:

Evaluación	Teoría
	TEORÍA 35% Examen Teórico E1 (35%) LABORATORIO 65% Evaluación Continua C1 (5%) Evaluación Continua C2 (5%) Práctica Calificada PC1 (15%) Práctica Calificada PC2 (15%) Práctica Calificada PC3 (15%) Proyecto P1 (10%) **Nota** : La ponderación de la evaluación se hará si ambas partes están aprobadas.
	100%

10. REFERENCIAS BIBLIOGRÁFICAS

[1] DEITEL, P., AND DEITEL, H. C++ How to Program. Prentice Hall, 2016.
[2] ECKEL, B. Thinking in C++, Vol. 1: Introduction to Standard C++. 2nd Edition, Prentice Hall,2000.
[3] STROUSTRUP, B. The C++ Programming Language. 4th. Addison-Wesley, 2017.