UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERIA
ESCUELA DE CIENCIAS Y SISTEMAS

Pensum de Estudios con Créditos CLAR
CARRERA DE INGENIERÍA EN CIENCIAS Y SISTEMAS

PROGRAMA DEL CURSO

NOMBRE DEL CURSO: INTRODUCCION A LA PROGRAMACION DE COMPUTADORES I - 0770

CODIGO:	0770	CREDITOS:	4
ESCUELA:	CIENCIAS Y SISTEMAS	AREA A LA QUE PERTENECE:	DESARROLLO DE SOFTWARE
PRE REQUISITO:	0768 INTRODUCCION A LOS ALGORITMOS Y FLUJOS DE DATOS	POST REQUISITO:	0771 INTRODUCCIÓN A LA PROGRAMACIÓN Y COMPUTACIÓN 2
CATEGORIA:	OBLIGATORIO	VIGENCIA:	PRIMER SEMESTRE 2026
CATEDRÁTICO (A):	VER ANEXO	AUXILIAR:	STAFF
EDIFICIO:	A DEFINIR	SECCIÓN:	A, B, C, D, E, F
SALÓN DEL CURSO:	A DEFINIR	SALON DEL LABORATORIO:	PENDIENTE
HORAS POR SEMANA DEL CURSO:	4	HORAS POR SEMANA DEL LABORATORIO:	2
DÍAS QUE SE IMPARTE EL CURSO:	DESCRITO EN INCISO 13	DIAS QUE SE IMPARTE EL LABORATORIO:	PENDIENTE
HORARIO DEL CURSO:	DESCRITO EN INCISO 13	HORARIO DEL LABORATORIO:	PENDIENTE

2.DESCRIPCIÓN DEL CURSO

El curso constituye la base primaria de programación del estudiante de la carrera de sistemas, a través del conocimiento de lenguajes, sus elementos base, el ciclo de desarrollo clásico del software; estructuras básicas para programar, manejo de memoria estática y dinámica, así como conceptos iniciales sobre lo que es la calidad y seguridad en el software. Dejando finalmente un punto inicial introductorio de lo que es la computación en la nube.

3. VINCULACIÓN DE COMPETENCIAS DEL PERFIL DE EGRESO

1. Demuestra pensamiento crítico, actitud investigativa y rigor analítico en el planteamiento y la resolución de problemas complejos.
2. Interpreta, analiza y aplica conceptos y procedimientos para la solución de problemas de ingeniería y ciencias afines por medio de actividades de aprendizaje asignadas.
3. Utiliza software actualizado como herramienta para modelar y resolver problemas de ingeniería y ciencias afines, a través de conocimientos y habilidades adquiridas en los cursos con la tecnología disponible.
4. Planifica y desarrolla actividades de auto aprendizaje para la solución de problemas por medio de la implementación de trabajos extra aula realizados de manera individual y/o grupal colaborativo.
5. Razona crítica y lógicamente sobre los procesos y resultados para verificar su validez por medio de la comparación con el conocimiento y la experiencia
6. Utiliza e interpreta el lenguaje natural y seudocódigo para la correcta comunicación y desarrollo de conocimiento científico, por medio de la redacción y lectura de publicaciones a nivel nacional e internacional.
7. Fortalece sus habilidades de trabajo individual y en equipo multidisciplinario para su buen desempeño profesional por medio de las actividades asignadas.

4. Unidad de Aprendizaje No 1: FUNDAMENTOS DE PROGRAMACION
Periodos: 02

Problema:
Que el estudiante comprenda que es un lenguaje de programación, tipos vigentes en el mercado, su estructura interna, las interfaces graficas donde corre; así como los elementos que lo constituyen y como estos permiten el trabajar con la memoria estática y dinámica, así como inspección de código.

Competencias de la unidad	• Identifica y analiza los fundamentos de un lenguaje de programación. • Conoce los elementos que conforman a un lenguaje de programación vigente. • Administra estructuras estáticas de memoria. • Ejecuta la inspección de código para garantizar el correcto funcionamiento de este.

Criterios de desempeño

Saber hacer	Saber conocer	Saber ser
Describe los fundamentos de un lenguaje Distingue paradigmas y los tipos de lenguajes de programación vigentes en el mercado. Identifica los elementos base de un lenguaje que son necesarios para el desarrollo de software. Reconoce la forma de implementar memoria estática y las estructuras asociadas a esta. Reconoce las estructuras dinámicas e implementación de esta en memoria. Inspecciona el código de diferentes formas para garantizar el funcionamiento del mismo, así como la inspección de errores en la ejecución.	Conoce que es un lenguaje; como se compone cuales son vigentes en el mercado. Diferencia la aplicación y uso de los diferentes paradigmas de programación. Identifica e implementa los elementos base en el lenguaje nativo designado en el curso. Diferencia e implementa estructuras estáticas o dinámicas según las necesidades en el desarrollo del software. Administra eficientemente la reserva de memoria del computador usando y colocando únicamente la estructura de datos necesarias para su código. Conoce como inspeccionar un código de forma eficiente identificando errores y mejoras en su ejecución.	Adquiere la habilidad de diferenciar los paradigmas y ventajas y desventajas que ofrecen según las necesidades. Resuelve una construcción de código implementando los elementos de cualquier lenguaje aprovechando para satisfacer las necesidades de negocio. Hace uso eficiente de la memoria en el desarrollo de una solución de negocio; en el lenguaje seleccionado; contribuyendo a un buen rendimiento de su construcción. Debugea errores en un código propio o ajeno; identifica mejoras en las construcciones y flujo lógicos en el código fuente de una solución. Provee soluciones fuertes en código a través de reducción de errores con una inspección del flujo de su código.

4.1 Evidencia de aprendizaje

- Tarea: Solución de ejercicios seleccionados del libro de texto de la Unidad 1, para trabajar individualmente en su casa.

4.2 Instrumento de Evaluación

Rúbricas de calificación de tareas (ver apartado de rúbricas al final del documento)

5. Unidad de Aprendizaje No 2: MEMORIA DINAMICA
Periodos: 03

Problema:
Entender la organización de la memoria y su administración dinámica dentro del ordenador. Su aplicación en soluciones de software y como las estructuras básicas diferenciadas permiten interactuar y almacenar información relevante en la ejecución de un programa.

Competencias de la unidad	• Identifica y analiza la estructura de memoria RAM, segmentación de esta y fines de cada uno de estos para almacenar diferentes tipos de datos. • Conce y construye punteros en nodos; aplicados a diferentes estructuras dinámicas de memoria que almacenan datos con diversas implementaciones eficientes para el manejo de datos. • Administra y proyecta el consumo de bytes en memoria RAM; planifica dentro de su programación el uso y liberación de memoria bajo demanda(memory allocation); así como uso del garbage collector.

Criterios de desempeño

Saber hacer	Saber conocer	Saber ser
Identifica que es la memoria dentro del ordenador y el papel que esta cumple en el manejo y traslado de información. Distingue los segmentos de memoria y donde se alojan los diferentes tipos de datos en la memoria. Identifica los elementos de un nodo y como estos deben ser asociados entre otros nodos como una construcción base. Reconoce estructuras dinámicas simples como listas y todas sus variaciones, pilas y colas. Reconoce otros tipos de estructuras como tablas de hash y arboles a un alto nivel sin mayor detalle como alternativas avanzadas de administración de la memoria. Identifica la cantidad de bytes a usar dentro de su codificación como base para la construcción de las estructuras dinámicas de su solución.	Sabe la importancia y el valor de la memoria en el ordenador, por que debe ser administrada correctamente Descompone como la memoria está estructurada e identifica que tipo de información es colocada en cada uno de los segmentos. Conoce los elementos y tipo de información que un nodo debe contener y como estos se conectan con otros. Identifica con facilidad las estructuras dinámicas vigentes y cuales son los casos de implementación Conoce estructuras dinámicas complejas a un alto nivel y los casos de implementación Cuantifica la cantidad de bytes iniciales de las estructuras y variables creadas en su código fuente y el consumo inicial que esta pueda tener al reservar memoria.	Adquiere la habilidad para crear únicamente las estructuras estáticas y dinámicas necesarias en memoria. Administra variables y objetos que son colocados en memoria proyectando los bytes iniciales en el segmento de datos(DS). Desarrolla un TDA implementando diferentes conexiones entre nodos y comprende la lógica de conexión. Adquiere la habilidad para implementar en un lenguaje una estructura dinámica Reconoce para que tipo de casos, una estructura dinámica compleja puede ser usada. Proyecta el consumo de bytes de las estructuras y variables creadas en su código y cual es su peso en el ordenador donde el código se ejecuta; así como evaluar oportunidades para optimizar el consumo de memoria RAM.

5.1 Evidencia de aprendizaje

- Tarea: Solución de ejercicios seleccionados Unidad 2, para trabajar individualmente en su casa.

5.2 Instrumento de Evaluación

Rúbricas de calificación de tareas (ver apartado de rúbricas al final del documento)

6. Unidad de Aprendizaje No 3: PROGRAMACION ORIENTADA A OBJETOS
Periodos: 02

Problema:
Comprender la programación orientada a objetos como parte del paradigma imperativo, de programación identificando los principios y como estos se concretan al programar, así como los procesos que permiten desarrollar una solución usando este tipo de programación.

Competencias de la unidad	• Identificar los principios de POO; y lo diferencia del resto de paradigmas o tipos de lenguaje. • Desde la abstracción hasta la diagramación de las clases el estudiante reconoce todo el proceso ordenado para la construcción del software. • Utiliza herramientas CAD/RAD para apoyarse en el proceso de construcción de SW usando POO. • Implementa soluciones entendiendo casos reales planteados para una construcción del software de forma en que todo lo organiza entorno a objetos existentes del lenguaje y otros creados por el estudiante.

Criterios de desempeño

Saber hacer	Saber conocer	Saber ser
Conoce y comprende cada uno de los principios de POO a nivel conceptual para posteriormente ser implementados. Distingue el proceso de POO desde la abstracción de objetos de una realidad hasta la creación del Diagrama de Clases. Identifica que previo al desarrollo debe existir un diseño (Blue Print) de este provisto para un lenguaje de POO a través el Diagrama de Clases. Conoce e identifica los objetos del lenguaje POO usado, así como construir nuevos objetos para su desarrollo. Identifica herramientas CAD/RAD para el diseño de clases y su fácil exportación de las clases y conexiones a un lenguaje fuente a elección.	Sabe el paso a paso de cada uno de los principios de POO y como estos se construyen en modelado hasta el desarrollo nativo del lenguaje Reconoce en la abstracción como el paso más importante para la construcción de clases en el mapeo de una realidad de negocio. Identifica claramente que el diseño de software tiene como componente principal la construcción y unión de clases. Conoce como se construye una clase; su nombre, atributos y métodos principales (constructor, destructor y getters y setters. Determina como se da vida o se instancia un objeto a través de forma única o combinado con otras estructuras previamente conocidas. Identifica herramientas CAD vigentes del mercado así como ventajas y desventajas asociadas.	Implementa nativamente en el lenguaje de desarrollo los principios de POO claramente enlazándolos todos en el producto final que es el Diagrama de Clases. Realiza los procesos de abstracción necesarios para mapear la realidad sus objetos e información clave sobre un realidad o problema de negocio dado. Identifica clases físicas/lógicas que lo llevan a la construcción de las mismas y unión respectiva. Crea clases con todos sus atributos y métodos necesarios; así como los tipos de conexiones, y multiplicidad de las mismas. Implementa la instancia de clases de forma simple o compuesta con las estructuras de datos previamente conocidas (estáticas o dinámicas de memoria). Construye código en un lenguaje POO usando como base una herramienta CAD/RAD vigente para eficientizar el desarrollo.

6.1 Evidencia de aprendizaje

- Tarea: Solución de ejercicios seleccionados Unidad 3, para trabajar individualmente en su casa.

6.2 Instrumento de Evaluación

Rúbricas de calificación de tareas (ver apartado de rúbricas al final del documento)

7. Unidad de Aprendizaje No 4: TESTING, SECURITY & QUALITY ASSURANCE I
Periodos: 04

Problema:
Reconocer e implementar procesos de construcción de calidad y seguridad del software para incluir estándares básicos del mercado vigentes.

Competencias de la unidad	• Reconoce los principios básicos de cómo realizar pruebas en el software, para garantizar la calidad de un proyecto de software. • Identifica los aspectos principales estándares de seguridad vigentes de software, reduciendo el riesgo de afectación de una solución. • Razona y establece prioridades generales que garanticen la elaboración del software con calidad, esto evitando afectaciones en la información y el software.

Criterios de desempeño

Saber hacer	Saber conocer	Saber ser
Determina que es una prueba de software, tipos y como es llevada a cabo de inicio a fin y sobre qué ambiente. Distingue como la prueba se asocia con los requerimientos funcionales y no funcionales del software. Desglosa los roles de prueba (Dev, QA y UAT), los ambientes y que tipos de prueba se realizan a lo largo de la vida del software. Reconoce estándares de calidad y seguridad vigentes en el mercado y como estos contribuyen al desarrollo. Se apoya de IAs orientadas al desarrollo de SW para validar y mejorar su código fuente con estándares de seguridad y calidad vigentes.	Identifica los tipos de pruebas alcances y como deben ser aplicadas para garantizar la funcionalidad del software. Sabe como implementar pruebas sobre la funcionalidad del software y aspectos no funcionales del mismo. Sabe cómo los roles de prueba del software impactan para garantizar el funcionamiento de este. Identifica estándares de QA y SA aplicables en los detalles relacionados con los elementos de desarrollo aprendidos incluyendo este curso conocido. Establece que estándares son los más usados en el mercado y cuales puede implementar asistidos por IA para la mejora de su código fuente; identificando los puntos de mejora.	Adquiere la habilidad de probar el software de formas distintas en diferentes etapas para garantizar el funcionamiento del mismo. Resuelve a través de un plan de pruebas, casos y escenarios garantizar que todos los requerimientos de negocio, así como técnicos funcionan correctamente. Adquiere la habilidad para coordinar pruebas en diferentes etapas, así como la construcción de los casos de pruebas para los diferentes roles de negocio. Entiende que los estándares de QA y SA continuamente se renuevan por vulnerabilidades o mejoras continuas y que es clave conocerlos para desarrollar software de calidad y seguro. Se apoya de la IA para validar la seguridad y calidad de su código fuente aplicando QA y SA vigente entendiendo lo que ha aplicado.

7.1 Evidencia de aprendizaje

- Tarea: Solución de ejercicios seleccionados Unidad 4, para trabajar individualmente en su casa

7.2 Instrumento de Evaluación

Rúbricas de calificación de tareas (ver apartado de rúbricas al final del documento)

10. Evaluación de Curso

Unidad de aprendizaje	Evidencia de aprendizaje	Instrumento evaluación	Fecha	Valoración
Unidad 1	Actividades, Investigaciones y Tareas de unidad	Tareas, Investigaciones y hojas de trabajo.		1.25 pts.
Unidad 1	Evaluación de rendimiento	Primer Examen Parcial		12 pts.
Unidad 2	Actividades, Investigaciones y Tareas de unidad	Tareas, Investigaciones y hojas de trabajo.		1.25 pts.
Unidad 2	Evaluación de rendimiento	Segundo Examen Parcial		13 pts.
Unidad 3	Actividades, Investigaciones y Tareas de unidad	Tareas, Investigaciones y hojas de trabajo.		1.25 pts.
Unidad 3	Evaluación de rendimiento	Tercer Examen Parcial		15 pts.
Unidad 4	Actividades, Investigaciones y Tareas de unidad	Tareas, Investigaciones y hojas de trabajo.		1.25 pts.
Unidad 4	Evaluación de rendimiento	Examen Final de Curso.		25pts.

11. Texto y referencias

- DEITEL, D. &.D (2008). Cómo Programar en Java (7ma Edición). Mexico: Prentice Hall.
- FREEMAN, E., ROBSON, E., & BATES, B. Y. (2009). Head First Design Patterns. USA: O'Reilly.
- JOYANES, L. (1995). Programación en Turbo Pascal Versiones 5.5, 6.0, y 7.0. Interamericana de España: McGraw-Hill Mexico.
- JOYANES, L. y. (2002). Programación en Java 2 (algoritmos, estructura de datos y programación orientada a objetos). España: McGraw-Hill / Interamericana de España, S. A. .
- MCLAUGHLIN, B., & POLLICE, G. Y. (2006). Head First Object-Oriented Analysis & Design. USA: O'Reilly Media.

12. Clausulas restrictivas

El perfil del estudiante de la facultad de Ingeniería de la Universidad de San Carlos de Guatemala exige una alta calidad en la excelencia académica y ética profesional. Se establecen en este curso los siguientes lineamientos que regulan el comportamiento del estudiante:

- Copias en exámenes, cortos, proyectos, tareas e investigaciones tienen cero de nota.
- Exámenes parciales y examen final NO tienen reposición.
- No hay prorrogas.
- No hay reposición de proyectos.
- Cualquier proyecto, tarea o investigación que se entregue después de la fecha calendarizada tiene 30 puntos menos, cada día de atraso.
- Los exámenes resueltos a lápiz no tienen derecho a revisión.
- Es obligatorio ganar el laboratorio para tener derecho a evaluación total del curso.
- 80% mínimo de asistencia.
- El curso se gana con 61 pts. de 100. El laboratorio de gana con 61 pts. de 100.

13. Dist.de clases

Introducción a la programación de Computadores I	A	S 07:10 - 10:30	Ing. Marlon Orellana
Introducción a la programación de Computadores I	B	M-J 07:10 - 08:50	Ing. William Argueta
Introducción a la programación de Computadores I	C	M-J 07:10 - 08:50	Ing. Moises Velasquez
Introducción a la programación de Computadores I	D	M-J 07:10 - 08:50	Ing. Herman Veliz
Introducción a la programación de Computadores I	E	M-J 07:10 - 08:50	Ing. Neftali Calderon
Introducción a la programación de Computadores I	F	L-M 11:30 - 13:10	Ing. William Escobar