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 II – 0771

| | | | |
|---|---|---|---|
| CODIGO: | 0771 | CREDITOS: | 4 |
| ESCUELA: | CIENCIAS Y SISTEMAS | AREA A LA QUE PERTENECE: | DESARROLLO DE SOFTWARE |
| PRE REQUISITO: | 0770<br>INTRODUCCION A LA PROGRAMACION DE COMPUTADORES I | POST REQUISITO: | 0773<br>ESTRUCTURA DE DATOS |
| CATEGORIA: | OBLIGATORIO | VIGENCIA: | SEGUNDO SEMESTRE 2026 |
| CATEDRÁTICO (A): | MBA. MSc. ING. CLAUDIA ROJAS MORALES DE MORÁN | AUXILIAR: | ISAÍ ELIEZER MAGDIEL MOLINA GUEVARA |
| EDIFICIO: | | SECCIÓN: | B |
| SALÓN DEL CURSO: | MEET | SALON DEL LABORATORIO: | MEET |
| HORAS POR SEMANA DEL CURSO: | 4 | HORAS POR SEMANA DEL LABORATORIO: | 2 |
| DÍAS QUE SE IMPARTE EL CURSO: | JUEVES Y VIERNES | DIAS QUE SE IMPARTE EL LABORATORIO: | MARTES |
| HORARIO DEL CURSO: | 7:10 – 8:50 horas | HORARIO DEL LABORATORIO: | 17:20 – 19:00 |

## 2.DESCRIPCIÓN DEL CURSO

El curso constituye una base intermedia a elementos de programación del estudiante de la carrera de sistemas, a través del conocimiento de memoria dinámica, TDAs comunes para implementar el uso de memoria a través de diferentes estructuras de datos, se enlistan también las arquitecturas base para desarrollo web así como conceptos intermedios sobre lo que es la calidad y seguridad en el software.

## 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: OTROS ALGORITMOS Y MEMORIA DINÁMICA APLICADA
Periodos: 2.5

**Problema:**
Que el estudiante comprenda los fundamentos base del análisis y diseño de algoritmos, complejidad en los algoritmos recursivos, algoritmos de ordenamiento e implementar el paradigma de divide y vencerás; concluyendo con la creación de matrices ortogonales/dispersas.

| | |
|---|---|
| **Competencias de la unidad** | • Conoce e implementa el análisis y diseño de algoritmos como una base para la resolución de problemas.<br>• Implementa algoritmos recursivos y de ordenamiento conociendo casos de uso de estos.<br>• Organiza algoritmos de resolución de ordenamiento y búsqueda bajo el paradigma divide y vencerás.<br>• Crea un TDA para implementar matrices dispersas usando memoria dinámica. |

**Criterios de desempeño**

| Saber hacer | Saber conocer | Saber ser |
|---|---|---|
| Describe algoritmos estándar para evaluar una estrategia antes de programar.<br><br>Describir cuantas operaciones hace un algoritmo en función del tamaño de los datos.<br><br>Prepara diferentes escenarios que se pueden suscitar al ejecutar un algoritmo.<br><br>Reconoce la eficiencia de los algoritmos que sirven para ordenar datos.<br><br>Estructura programas fragmentados bajo el principio divide y vencerás para resolver problemas<br><br>Crea una estructura compleja de cuatro punteros para representar una matriz de nodos de memoria dinámica. | Conoce los componentes estructurales base de un algoritmo y como puede generar eficiencia en el mismo.<br><br>Mide y monitorea el tiempo, eficiencia y ejecución de algoritmos comparándolos a través de la notación Big O.<br><br>Identifica e implementa correctamente la recursividad en algoritmos en casos de usos prácticos.<br><br>Diferencia el tipo de algoritmo de ordenamiento necesario en base a su eficiencia acorde a variables de entorno relacionadas a datos.<br><br>Identifica que para resolver un problema el principio divide y vencerás provee un marco de trabajo para resolver problemas grandes o complejos<br><br>Conoce como se implementa en memoria la matriz ortogonal/dispersa. | Adquiere la habilidad para entender la estructura de un algoritmo, así como hacerlo eficiente en su ejecución.<br><br>Implementa de forma adecuada algoritmos eficientes en su ejecución y tiempo.<br><br>Conoce como y cuando implementar recursividad acorde a las necesidades de cualquier problema.<br><br>Implementa diversos tipos algoritmos de ordenamiento: considerando los GPU's del procesador, tamaño de la memoria y. volumen y tipo de datos.<br><br>Deconstruye un problema en componentes accionables que permitan la resolución de este.<br><br>Organiza bloques de nodos con cuatro punteros entrelazados para formar una estructura dinámica matricial de dos dimensiones. |

### 4.1 Evidencia de aprendizaje

- Tarea: Solución de ejercicios seleccionados 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: ESTRUCTURAS NO LINEALES BASICAS
Periodos: 03

**Problema:**
Apoyar al estudiante a profundizar en el uso de estructuras no lineales para organizar información de forma jerárquica no línea; permitiendo que los programas crezcan o se reduzcan de tamaño eficientemente en el tiempo; a través de la utilización de punteros. .

| | |
|---|---|
| **Competencias de la unidad** | • Identifica y analiza que es la estructura no lineal como un árbol; como este se implementa en memoria, operaciones base asociadas a este y ventajas asociadas.<br>• Conce y construye punteros en nodos organizados en forma no lineal; a través de construcciones de aboles binarios y AVL respectivamente.<br>• Diseña y construye los TDAs jerárquicos, con objetivos específicos para administrar datos organizados no linealmente de forma balanceada o no; con sus debidas operaciones asociadas. |

**Criterios de desempeño**

| Saber hacer | Saber conocer | Saber ser |
|---|---|---|
| Comprende las estructuras no lineales de almacenamiento dinámico y su representación gráfica.<br><br>Distingue el modelado de objetos aplicados a estructuras dinámicas como árboles.<br><br>Identifica las relaciones entre nodos y como se diseñan estos para una representación no lineal.<br><br>Reconoce casos de uso para implementar estructuras como arboles binarios y AVL.<br><br>Reconoce las operaciones asociadas a los arboles binarios y AVL respectivamente<br><br>Identifica la diferencia entre arboles balanceados y no balanceados y la comparativa de eficiencia entre uno y otros. | Sabe la importancia de implementar estructuras no lineales para el manejo de datos eficientemente.<br><br>Conoce como a través del uso de objetos la construcción de árboles puede ser implementada.<br><br>Conoce como diseñar los TDAs de estructuras no lineales y sus correspondientes relaciones.<br><br>Identifica los casos de uso donde los arboles sean la solución idónea para el manejo de datos.<br><br>Conoce la lógica de las operaciones de las estructuras no lineales de los árboles binarios y AVL.<br><br>Evalúa como deben ser recorridos y operados los árboles; reconociendo la complejidad y beneficios de uno sobre otro. | Adquiere la habilidad de diseñar estructuras no lineales para el manejo eficiente de datos en memoria.<br><br>Administra a través de objetos, construcciones no lineales a través memoria dinámica.<br><br>Desarrolla y entiende los TDA de árboles binarios y AVL correctamente para ser implementados en lenguajes de programación.<br><br>Adquiere habilidades, para identificar los casos donde la implementación de una estructura no lineal es útil y eficiente.<br><br>Reconoce e implementa en un lenguaje de programación vigente o no, las operaciones de un árbol binario y un AVL respectivamente.<br><br>Evalúa los beneficios de implementar arboles en una solución de negocio y sus ventajas |

### 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: ARQUITECTURAS PARA SOLUCIONES WEB
Periodos: 08

**Problema:**
Comprender los elementos arquitectónicos base vigentes, para la construcción de una solución web end-to-end. Desde la notación de diseño estándar para diagramarlos, los modelos de despliegue, conceptos de desarrollo, patrones de diseño comunes, una construcción básica web hasta como desplegarla ente los diferentes ambientes (Dev, QA, Prod ).

| | |
|---|---|
| **Competencias de la unidad** | • Identificar los dos diagramas UML base para desplegar una solución de software.<br>• Distingue los modelos arquitectónicos vigentes, donde puede desplegar el software.<br>• Implementa los elementos y lógica de desarrollo web usando tecnologías vigentes.<br>• Diseña e implementa los elementos de capas en el patrón MVC para una solución web.<br>• Construye los elementos base para una aplicación web; integra componentes y comprende como se despliega la misma en los diferentes ambientes. |

**Criterios de desempeño**

| Saber hacer | Saber conocer | Saber ser |
|---|---|---|
| Conoce y comprende los diagramas UML que denotan los componentes de una solución y como estos son desplegados en los ambientes del ciclo de vida de esta.<br><br>Distingue los diferentes modelos de arquitectura de software donde puede ser desplegada una solución.<br><br>Identifica que elementos necesita para el desarrollo de una solución web y como integrarlos de forma segura y funcional.<br><br>Comprende la lógica y el diseño de la arquitectura de software MVC que separa la lógica de una aplicación, los datos y la interfaz de usuario.<br><br>Identifica los componentes base de una aplicación web, la integración de componentes y finalmente como se despliega entre ambientes. | Sabe como usar los Diagramas UML de componentes y despliegue para implementar una solución de software.<br><br>Reconoce ante una necesidad de negocio el modelo de arquitectura idóneo para implementar una solución de software.<br><br>Conoce como y en que herramientas puede ser implementados cada uno de los elementos de una solución de software<br><br>Diferencia la forma en que se separan y se comunican e implementan las capas de una solución basada en la arquitectura MVC.<br><br>Determina que elementos debe usar en las capas de negocio, datos e interfaz de usuario; así como los aspectos de seguridad de estas capas.<br><br>Conoce herramientas para la construcción de los elementos de una solución web y su integración y el orden de despliegue del software entre ambientes. | Diseña con herramientas CAD/RAD vigentes los diagramas de componentes y despliegue de una solución de Software.<br><br>Implementa una solución end to end sobre un modelo arquitectónico de software acorde a las necesidades y análisis del negocio.<br><br>Utiliza herramientas vigentes en el mercado para la construcción e integración de los elementos de una solución web.<br><br>Crea e integra los elementos de cada una de las capas de una solución de una forma segura usando el patrón MVC.<br><br>Aplica en la integración de las capas de una solución que usa el patrón MVC buenas prácticas de acople y seguridad estándar.<br><br>Utiliza herramientas vigentes para hacer el deploy de una solución de software en los ambientes Dev, QA y Prod respectivamente de forma segura y ordenada. |

### 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: 3.5

**Problema:**
Guiar al estudiante a la elaboración de un set estándar mínimas de pruebas de software; desde su definición hasta su ejecución, manual o automáticamente; usando la IA como asistencia para la mejora de la calidad y seguridad del software.

| | |
|---|---|
| **Competencias de la unidad** | • Entiende con claridad que son las pruebas de software y los tipos asociados.<br>• Construye casos de prueba, los ejecuta para identificar errores y corregirlos para garantizar funcionalidad y seguridad del software.<br>• Maneja herramientas de apoyo para automatizar las pruebas a través de flujos de trabajo.<br>• Implementa buenas prácticas de seguridad en el desarrollo basado en secciones base de OWAsp; orientado a las aplicaciones Web y Moviles respectivamente. |

**Criterios de desempeño**

| Saber hacer | Saber conocer | Saber ser |
|---|---|---|
| Determina que es una prueba de software, tipos asociados; manejos de estas en entornos agiles de desarrollo.<br><br>Implementa diversos tipos de pruebas en entornos agiles de desarrollo analizando la funcionalidad y código.<br><br>Identifica que aspectos deben ser probadas en las entradas y salidas del software; así como debe de aplicarse el correcto tratamiento de errores.<br><br>Reconoce el estándar OWASP y su contribución a los aspectos de seguridad y desarrollo del software.<br><br>Implementa dentro del ciclo de desarrollo del software buenas prácticas end to end a cada una de las etapas para garantizar la calidad y seguridad de este. | Conoce como aplicar acorde a los aspectos funcionales o no funcionales del software las pruebas correspondientes.<br><br>Sabe cuándo y cómo implementar pruebas unitarias, de integración n de seguridad, agile y shift left respectivamente.<br><br>Sabe que componentes de software impactan y deben ser probados para garantizar el funcionamiento de este.<br><br>Identifica los elementos de prácticas de seguridad y desarrollo OWASP en su última versión que puede ser usada para el desarrollo de una solución web.<br><br>Establece en cada una de las etapas del ciclo de desarrollo las buenas prácticas mínimas en la construcción del software, así como en los ambientes en el que es desplegado el software. | Utiliza herramientas vigentes para implementar un plan de pruebas sobre el software.<br><br>Resuelve usando un plan de pruebas el tipo de pruebas seleccionadas usando software y herramientas de IA.<br><br>Adquiere la habilidad para detectar errores frecuentes, en el software de forma asistida o no asistida.<br><br>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.<br><br>Se apoya de las buenas prácticas de seguridad y calidad de software usando OWASP top 10 estándar para aplicaciones web. |

### 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. | | 2.5 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. | | 2.5 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. | | 2 pts. |
| Unidad 3 | Evaluación de rendimiento | Tercer Examen Parcial | | 13 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


- Introducción al análisis y diseño de algoritmos (2014).Maria Gomez Fuentes
- Fundamentos de programación, Algoritmos, Estructuras de Datos y Objetos, Luis Joyanes Aguilar, Cuarta Edición, McGraw-Hill
- Craig Larman; Introducción al análisis y diseño orientado a objetos. Prentice Hall
- Metrics for software conceptual models; (Editores: Marcela Genero, Mario Piattini, Coral Calero,
- Java y el patron Modelo Vista y Controlador (2019); Sonia Alexandra Pinzo Nuñez ISBN 978-958-787-491-4
- Software Quality Assurance, From Theory to Implementation, Daniel Galin 2004.
- State of Sofware Development, CodingSans 2019
- El control de versions, Guillem Borrell, 2006.
- International Software Testing Qualifications Board. (2021). Agile Tester syllabus. ISTQB®. https://www.istqb.org/
- Software Design, David Budgen, 2ª. Edición, Pearson Addison Wesley.
- OWASP Application Security Verification Standard 4.0, 2019


## 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, plagio o uso de IA en exámenes, cortos, proyectos, tareas e investigaciones tienen cero de nota.
- Exámenes parciales y examen final NO tienen reposición.
- No hay prorrogas ni reposición de proyectos.
- Cualquier tarea o investigación entregada después de la fecha tiene 30 puntos menos por 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 para tener derecho a examen final.
- El curso se gana con 61 pts. de 100. El laboratorio de gana con 61 pts. de 100.
- Habrá 3 proyectos de programación.
- El catedrático revisará las notas obtenidas en el curso y el laboratorio. Podrá decidir sí es necesaria una segunda revisión a cada proyecto y considerar nuevamente la ponderación obtenida en cada proyecto.
- Las notas de cada proyecto serán publicadas por el catedrático del curso en el transcurso del semestre, el estudiante tendrá 8 días como máximo para pedir revisión de proyecto.
- Es obligatorio ganar el laboratorio para tener derecho a evaluación final del curso.
- No habrá proyecto de retrasada, ni reposición de nota de laboratorio.

## 13. Dist.de clases

| | | | |
|---|---|---|---|
| Introducción a la programación de Computadores II | A | J-V 07:10 - 08:50 | MARLON A. PEREZ TURK |
| Introducción a la programación de Computadores II | B | J-V 07:10 - 08:50 | CLAUDIA L. ROJAS MORALES |
| Introducción a la programación de Computadores II | C | J-V 07:10 - 08:50 | JOSE M. RUIZ JUAREZ |
| Introducción a la programación de Computadores II | D | V 07:10 - 10:30 | DENNIS BARRIOS GONZALES |
| Introducción a la programación de Computadores II | N | J-V 07:10 - 08:50 | EDWIN E. ZAPETA GOMEZ |
| Introducción a la programación de Computadores II | P | L-M 17:20 - 19:00 | FERNANDO J. PAZ GONZALES |