1. DATOS GENERALES DE LA UNIDAD DE APRENDIZAJE (UA) O ASIGNATURA			
Nombre de la Unidad de Aprendizaje (UA) o Asignatura			**Clave de la UA**
SEMINARIO DE PROBLEMAS DE PROGRAMACIÓN			I9888
Modalidad de la UA	**Tipo de UA**	**Área de formación**	**Valor en créditos**
Presencial	Seminario	Básica Particular	4

UA de pre-requisito	UA simultáneo	UA posteriores
Ninguna	Se sugiere cursar Programación	No tiene seriación para las carreras de INRO y IGFO

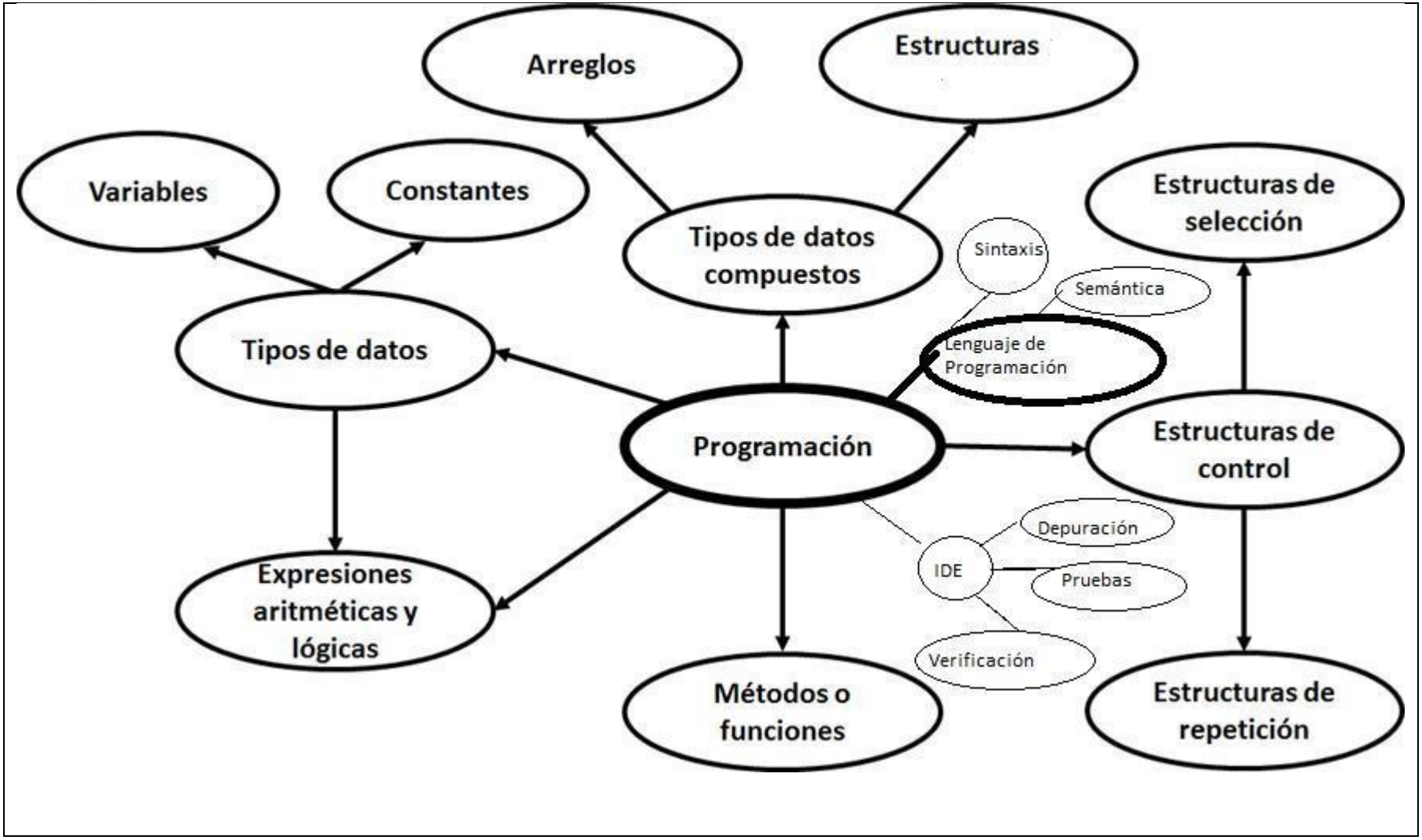
Horas totales de teoría	Horas totales de práctica	Horas totales del curso
0	68	68

Licenciatura(s) en que se imparte	Módulo al que pertenece
Ingeniería Robótica Ingeniería Fotónica	
Departamento	**Academia a la que pertenece**
Ciencias Computacionales	Programación
Elaboró	**Fecha de elaboración o revisión**
	JULIO 2023

<table>
<tr><th colspan="3">2. DESCRIPCIÓN DE LA UA O ASIGNATURA</th></tr>
<tr><th colspan="3">Presentación</th></tr>
<tr><td colspan="3">Este seminario aporta al perfil de los Ingenieros en Robótica y Fotónica, los conocimientos, habilidades, metodología, así como capacidades de análisis y síntesis, para plantear soluciones de problemas codificando programas que respondan a la solución de problemas utilizando el lenguaje de programación C.</td></tr>
<tr><th colspan="3">Relación con el perfil</th></tr>
<tr><th colspan="3">Competencias a desarrollar en la UA o Asignatura</th></tr>
<tr><th>Transversales</th><th>Genéricas</th><th>Profesionales</th></tr>
<tr>
<td>● Capacidad de abstracción, análisis y síntesis
● Identificar y resolver problemas
● Comprensión y construcción de procesos
● Capacidad de crítica y autocrítica
● Análisis de la realidad
● Toma de decisiones
● Capacidad de comunicación oral y escrita
● Motivar y conducir hacia metas comunes
● Trabajo en equipo y colaborativo
● Capacidad de aplicar conocimientos en la práctica
● Capacidad de organizar el tiempo
● Capacidad de actuar en nuevas situaciones
● Capacidad de aprender y actualizarse
● Trabajo autónomo
● Capacidad creativa</td>
<td>● Identifica de un problema dado los elementos necesarios para su solución utilizando un lenguaje de programación estructurado.
● Estructura la solución del problema en módulos individuales programables.
● Codifica, revisa, y ejecuta la solución del problema y subproblemas utilizando buenas prácticas de programación.</td>
<td>● Analiza, diseña, desarrolla e implementa soluciones relacionados con hardware y software.</td>
</tr>
</table>

● Solidaridad ● Habilidades interpersonales		
Saberes involucrados en la UA o Asignatura		
Saber (conocimientos)	**Saber hacer (habilidades)**	**Saber ser (actitudes y valores)**
● Elementos del paradigma de programación Estructurada. ● Características de un compilador de Lenguaje Estructurado como herramienta para la codificación de la solución del problema ● Lenguaje de programación Estructurado	● Selecciona la solución de problemas utilizando la abstracción. ● Usa un lenguaje de programación estructurado y un respectivo IDE. ● Usa equipos de cómputo. ● Integra los conceptos básicos de la Programación Estructurada en la solución de problemas	● Respeto a la agenda propuesta para la materia: puntualidad y asistencia. ● Entrega de trabajo originales. ● Sentido crítico y análisis grupal a las soluciones individuales del grupo. ● Fomento a la iniciativa entre los alumnos del grupo.
Producto Integrador Final de la UA o Asignatura		
Título del Producto: Análisis, diseño y codificación de un sistema. **Objetivo**: Aplicar el paradigma de programación estructurada en un problema dado. **Descripción**: La solución al problema debe contener las siguientes características: ● Análisis: identificar los elementos del problema y sus relaciones. ● Diseño de la solución: mediante la utilización de diagramas y / o pseudocódigos. ● Implementación: codificar considerando buenas prácticas de programación. La solución debe ser modular e incluir estructuras selectivas, repetitivas, datos estructurados como arreglos y registros. ● Funcionalidad: el programa debe cumplir con las especificaciones establecidas.		

3. ORGANIZADOR GRÁFICO DE LOS CONTENIDOS DE LA UA O ASIGNATURA



4. SECUENCIA DEL CURSO POR UNIDADES TEMÁTICAS
Unidad temática 1: Conceptos básicos del Lenguaje C

Objetivo de la unidad temática: Aplicar los conceptos básicos de programación utilizando el entorno del compilador para la solución de un problema.

Introducción: En esta unidad se describirán los conceptos básicos del lenguaje de programación C. Mediante el empleo de entornos de desarrollo, el alumno editará, compilará y ejecutará sus primeros programas. Estos primeros programas servirán de base en unidades posteriores.

Contenido temático	Saberes involucrados	Producto de la unidad temática
1.1 Historia del lenguaje C 1.2 Entorno del compilador 1.3 Elementos de un programa 1.3.1 Identificador 1.3.2 Tipos de datos primitivos 1.3.3 Definición y declaración de datos 1.3.3.1 Variables 1.3.3.2 Constantes (const, #define) 1.3.4 Operadores y expresiones 1.3.5 Palabras reservadas 1.3.6 Comentarios	Identifica y utiliza elementos básicos de programación tales como variables, constantes y operadores en procesos de entrada-salida, así como en expresiones para resolver problemas. Utiliza un lenguaje de programación para codificar las soluciones de los problemas. Identifica y organiza los elementos que se requiere para resolver un problema. Elabora y evalúa expresiones matemáticas simples, aplicando la correcta prioridad de los operadores. Presenta sus productos en tiempo y forma, de tal manera que demuestra interés y cuidado en su trabajo. Fortalece el trabajo colaborativo y en equipo.	Archivo .c con la codificación de por lo menos un programa que involucre el uso de variables, constantes y operadores. Donde exista la entrada, proceso y salida de datos. Bajo las siguientes características: ● Inicializar siempre las variables cuando se declaran. ● Expresar valores literales como constantes.

Actividades del docente	Actividades del estudiante	Evidencia de la actividad	Recursos y materiales	Tiempo destinado

Solicita a los estudiantes lecturas previas de la historia del lenguaje C.	Identificar las razones que llevaron a la creación de este lenguaje, así como su evolución.	Reporte (archivo en Word o pdf)	Internet y / o bibliografía sugerida.	1
Pide a los estudiantes la instalación de un entorno de desarrollo adecuado para el lenguaje C. Ejemplifica el uso del IDE para la codificación de un primer programa.	Aprender a utilizar las funciones básicas del entorno como crear archivo nuevo, guardar cambios, compilar y ejecutar.	Archivo .c de un primer programa	PC y el IDE (Entorno de Desarrollo Integrado)	2
El profesor indica a los alumnos organizarse en grupos para resolver una serie de ejercicios donde se apliquen los conceptos básicos. El profesor aclara las dudas de los grupos de trabajo.	Los alumnos discutirán la mejor manera para resolver los problemas. Los alumnos evaluarán expresiones aritméticas y lógicas presentadas por el profesor.	Código (archivos .c) de los programas funcionales que resuelven los problemas propuestos	PC y el IDE	5

Unidad temática 2: Programación Estructurada

Objetivo de la unidad temática: Crear programas utilizando el paradigma de Programación Estructurada.

Introducción: En esta unidad se explicarán las estructuras de control que maneja el lenguaje de programación C. Mediante el empleo del entorno de desarrollo instalado, el alumno desarrollará los programas que involucren estos temas. Estos programas servirán de base en unidades posteriores.

Contenido temático	Saberes involucrados	Producto de la unidad temática
2.1 Estructuras de control 2.1.1 Clasificación de estructuras selectivas 2.1.1.1 Secuencial 2.1.1.2 Selectiva simple (if) 2.1.1.3 Selectiva doble (if - else) 2.1.1.4 Selectiva múltiple (switch) 2.2 Estructuras de control repetitivas o de iteración condicional	Clasifica y aplica las diferentes estructuras de control en los problemas a resolver. Usa de manera correcta contadores, acumuladores y banderas. Usa de manera correcta los operadores lógicos y relacionales en las expresiones	Archivos .c de cada una de las estructuras de control, aplicando el uso de banderas, contadores y acumuladores. Utilizando buenas prácticas de programación como:

2.2.1 Contadores, acumuladores y banderas 2.2.2 Mientras (while) 2.2.3 Hacer mientras (do - while) 2.2.4 Desde (for) 2.3 Implementación de estructuras anidadas	(condiciones). Fortalece la abstracción en la solución de problemas. Mejora sus habilidades de comunicación oral y escrita.	● Inicializar siempre las variables cuando se declaran. ● Expresar valores literales como constantes. ● Uso de sangría adecuado. ● Usar siempre llaves ({}) en las estructuras de control. ● Colocar al lado de una llave que cierre un bloque de código un indicativo de que tipo de estructura cierra.

Actividades del docente	**Actividades del estudiante**	**Evidencia de la actividad**	**Recursos y materiales**	**Tiempo destinado**
Solicita a los estudiantes lecturas previas acerca de las estructuras selectivas: if, if - else, switch El profesor resuelve las dudas que tuvieron los estudiantes en la resolución de problemas con las sentencias selectivas. El profesor proporciona una serie adicional de ejercicios a resolver para fortalecer el uso de las estructuras selectivas.	Resolverá problemas en los que aplique sus conocimientos sobre las tres estructuras selectivas.	Códigos (archivos .c) de por lo menos un programa por cada una de las estructuras selectivas.	Internet y / o bibliografía sugerida. PC e IDE	10
Solicita a los estudiantes lecturas previas acerca de las estructuras	Resolverá problemas en los que aplique sus conocimientos sobre las estructuras	Códigos (archivos .c) de por lo	Internet y / o bibliografía	10

repetitivas: while, do-while, for. El profesor resuelve las dudas que tuvieron los estudiantes en la resolución de problemas con las estructuras repetitivas. El profesor proporciona una serie adicional de ejercicios a resolver para fortalecer el uso de las estructuras repetitivas.	repetitivas.	menos un programa funcional por cada una de las estructuras.	sugerida. PC e IDE	
El profesor planteará una serie de ejercicios con mayor grado de complejidad a resolver con estructuras anidadas, tanto selectivas como repetitivas. Organiza un debate con las diferentes soluciones a la serie de ejercicios.	El alumno codificará las soluciones a una serie de ejercicios propuestos donde aplique las estructuras anidadas. Presenta y justifica sus propuestas de soluciones y discute con sus compañeros de grupo sobre las ventajas y desventajas de las diferentes soluciones con respecto a las demás presentadas en el grupo.	Códigos (archivos .c) de por lo menos un programa funcional por cada una de las estructuras. El archivo debe de estar debidamente comentado.	Internet y / o bibliografía sugerida. PC e IDE Cañón.	4

Unidad temática 3: Arreglos

Objetivo de la unidad temática: Crear programas utilizando los diferentes tipos de arreglos de datos.

Introducción: En esta unidad se utilizarán los diferentes tipos de arreglos que maneja el lenguaje de programación C. Mediante el empleo del entorno de desarrollo instalado, el alumno desarrollará los programas que involucren estos temas. Estos programas servirán de base en unidades posteriores.

Contenido temático	Saberes involucrados	Producto de la unidad temática
3.1 Definición 3.2 Tipos de arreglos 3.2.1 Vectores 3.2.2 Matrices	Clasifica y aplica de manera adecuada los diferentes tipos de arreglos en los problemas a resolver.	Archivos .c por cada tipo de dato presentar un programa tanto en vectores como matrices (al menos 6 programas).

	Presenta sus productos en tiempo y forma, de tal manera que demuestra interés y cuidado en su trabajo Refuerza la práctica del uso de las estructuras selectivas y repetitivas. Aplica operaciones sobre conjuntos de datos: ordena, analiza, promedia, entre otros. Aplica conocimientos en la práctica. Practica la toma de decisiones.	Utilizando buenas prácticas de programación como: ● Inicializar siempre las variables cuando se declaran. ● Expresar valores literales como constantes. ● Uso de sangría adecuado. ● Usar siempre llaves ({}) en las estructuras de control. ● Colocar al lado de una llave que cierre un bloque de código un indicativo de que tipo de estructura cierra.

Actividades del docente	**Actividades del estudiante**	**Evidencia de la actividad**	**Recursos y materiales**	**Tiempo destinado**
Solicita a los estudiantes lecturas previas acerca de arreglos: vectores y matrices. El profesor planteará una serie de ejercicios con mayor grado de complejidad a resolver con arreglos, tanto vectores como matrices. Organiza un debate con las diferentes soluciones a la serie de ejercicios.	El alumno codificará las soluciones a una serie de ejercicios propuestos donde aplique los arreglos. Presenta y justifica sus propuestas de soluciones y discute con sus compañeros de grupo sobre las ventajas y desventajas de las diferentes soluciones con respecto a las demás presentadas en el grupo	Códigos (archivos .c) de por lo menos un programa funcional por cada uno de los tipos de arreglos. El archivo debe de estar debidamente comentado.	Internet y / o bibliografía sugerida. PC e IDE Cañón.	8
Unidad temática 4: Manejo de funciones				

Objetivo de la unidad temática: Crear programas aplicando la programación modular.

Introducción: En esta unidad se aplicará el principio de "divide y vencerás" en la solución de problemas utilizando los distintos tipos de funciones que maneja el lenguaje de programación C. Mediante el empleo del entorno de desarrollo instalado, el alumno desarrollará los programas que involucren estos temas. Estos programas servirán de base en unidades posteriores

Contenido temático	Saberes involucrados	Producto de la unidad temática
4.1 Definición 4.2 Tipos de funciones 4.2.1 Funciones sin paso de parámetros 4.2.2 Funciones con paso de parámetros 4.2.2.1 Funciones con parámetros (tipos de datos primitivos) 4.2.2.2 Funciones con parámetros (tipo arreglo) 4.2.3 Funciones predefinidas 4.2.3.1 Matemáticas 4.2.3.2 De manejo de caracteres	Analiza y diseña una solución descendente de un problema dado. Descompone el problema en subproblemas. Codifica los subproblemas como funciones. Decide qué tipo de función implementa de acuerdo a cada subproblema. Usa adecuadamente variables locales y globales en la solución de sus problemas. Presenta sus productos en tiempo y forma, de tal manera que demuestra interés y cuidado en su trabajo	Codificación de programas modulares que resuelvan problemas sencillos y que involucren el uso de los distintos tipos de funciones. Archivos .c por cada tipo de dato presentar un programa tanto en vectores como matrices (al menos 6 programas). Utilizando buenas prácticas de programación como: ● Inicializar siempre las variables cuando se declaran. ● Expresar valores literales como constantes. ● Usar de manera adecuada las variables globales y locales. ● Evitar el uso de sentencias goto, break. ● Usar un único return por función, que se

			colocará como última sentencia de la función. ● Evitar escribir funciones y procedimientos demasiado largos. ● Evitar copiar y pegar segmentos casi idénticos de código un mismo programa. ● Usar siempre llaves ({}) en las estructuras de control. ● Colocar al lado de una llave que cierre un bloque de código un indicativo de que tipo de estructura cierra.	
Actividades del docente	**Actividades del estudiante**	**Evidencia de la actividad**	**Recursos y materiales**	**Tiempo destinado**
Solicita a los estudiantes lecturas previas acerca de funciones en C. El profesor planteará una serie de ejercicios con mayor grado de complejidad a resolver con todos los tipos de funciones. Organiza un debate con las diferentes soluciones a la serie de ejercicios.	El alumno codificará las soluciones a una serie de ejercicios propuestos donde aplique los diferentes tipos de funciones. Presenta y justifica sus propuestas de soluciones y discute con sus compañeros de grupo sobre las ventajas y desventajas de las diferentes soluciones con respecto a las demás presentadas en el grupo	Códigos (archivos .c) de por lo menos un programa funcional por cada uno de los tipos de funciones. El archivo debe de estar debidamente comentado.	Internet y / o bibliografía sugerida. PC e IDE Cañón.	16
Unidad temática 5: Registros				

Objetivo de la unidad temática: Crear programas en donde se requiera el diseño de datos compuestos complejos a partir de los datos simples y arreglos de datos

Introducción: En esta unidad se aplicará el uso de registros para manipular la información de una forma organizada. Mediante el empleo del entorno de desarrollo instalado, el alumno desarrollará los programas que involucren estos temas. Estos programas servirán de base para UA posteriores.

Contenido temático	Saberes involucrados	Producto de la unidad temática
5.1 Definición 5.2 Operaciones con registros 5.2.1 Entrada de datos de los elementos de un registro. 5.2.2 Salida de datos de un registro 5.3 Arreglos de registros 5.3.1 Definición 5.3.2 Operaciones con arreglos de registros 5.3.2.1 Entrada de datos de los elementos de un arreglo de registros 5.3.2.2 Salida de datos de un arreglo de registros.	Abstrae el problema organizando la información en tipos de datos compuestos complejos. Presenta sus productos en tiempo y forma, de tal manera que demuestra interés y cuidado en su trabajo Refuerza la práctica del uso de las estructuras selectivas y repetitivas. Aplica operaciones sobre conjuntos de datos: ordena, analiza, promedia, entre otros. Aplica conocimientos en la práctica. Practica la toma de decisiones.	Codificación de programas modulares que resuelvan problemas sencillos y que involucren el uso de registros de datos. Archivos .c con registros (al menos 1 programa). Utilizando buenas prácticas de programación como: ● Inicializar siempre las variables cuando se declaran. ● Expresar valores literales como constantes. ● Usar de manera adecuada las variables globales y locales. ● Evitar el uso de sentencias goto, break. ● Usar un único return por función, que se

			colocará como última sentencia de la función. ● Evitar escribir funciones y procedimientos demasiado largos. ● Evitar copiar y pegar segmentos casi idénticos de código un mismo programa. ● Usar siempre llaves ({}) en las estructuras de control. ● Colocar al lado de una llave que cierre un bloque de código un indicativo de que tipo de estructura cierra.	
Actividades del docente	**Actividad del estudiante**	**Evidencia de la actividad**	**Recursos y materiales**	**Tiempo destinado**
Solicita a los estudiantes lecturas previas acerca de registros. El profesor planteará por lo menos un ejercicio a resolver con registros en el cual se apliquen todas las unidades temáticas previas. Organiza una exposición de las diferentes soluciones que hubieran realizado los alumnos.	El alumno codificará la solución al ejercicio propuesto donde aplique registros. Expone y justifica su propuesta de solución, discute con sus compañeros de grupo sobre las ventajas y desventajas de la solución con respecto a las demás presentadas en el grupo	Código (archivo .c) de por lo menos un programa funcional que incluya el uso de registros. El archivo debe de estar debidamente comentado.	Internet y / o bibliografía sugerida. PC e IDE Cañón.	12

5. EVALUACIÓN Y CALIFICACIÓN

Requerimientos de acreditación:

De acuerdo al "REGLAMENTO GENERAL DE EVALUACIÓN Y PROMOCIÓN DE ALUMNOS DE LA UNIVERSIDAD DE GUADALAJARA":

Artículo 5. "El resultado final de las evaluaciones será expresado conforme a la escala de calificaciones centesimal de 0 a 100, en números enteros, considerando como mínima aprobatoria la calificación de 60."

Artículo 20. "Para que el alumno tenga derecho al registro del resultado final de la evaluación en el periodo ordinario, establecido en el calendario escolar aprobado por el H. Consejo General Universitario, se requiere:

I. Estar inscrito en el plan de estudios y curso correspondiente, y

Tener un mínimo de asistencia del 80% a clases y actividades registradas durante el curso."

De acuerdo al "REGLAMENTO GENERAL DE EVALUACIÓN Y PROMOCIÓN DE ALUMNOS DE LA UNIVERSIDAD DE GUADALAJARA":

Artículo 27. "Para que el alumno tenga derecho al registro de la calificación en el periodo extraordinario, se requiere:
I. Estar inscrito en el plan de estudios y curso correspondiente.
II. Haber pagado el arancel y presentar el comprobante correspondiente.
III. Tener un mínimo de asistencia del 65% a clases y actividades registradas durante el curso."

Criterios generales de evaluación:

Esta UA requiere de la presentación de programas en Lenguaje C, los cuales deben cumplir con lo siguiente:

- Entrega en tiempo
- Los códigos deben estar comentados con datos de quien lo elabora, la fecha de elaboración, descripción del programa.
- Queda estrictamente prohibido el plagio

Para la exposición el alumno debe mostrar plena compresión del contenido de su proyecto y responder satisfactoriamente a los cuestionamientos orales que se le hagan del mismo.

Evidencias o Productos			
Evidencia o producto	**Competencias y saberes involucrados**	**Contenidos temáticos**	**Ponderación**
1. Archivo .c con la codificación de por lo menos un programa que involucre el uso de variables, constantes y operadores. Donde exista la entrada, proceso y salida de datos. Bajo las siguientes características: ● Inicializar siempre las variables cuando se declaran. ● Expresar valores literales como constantes.	Identifica y utiliza elementos básicos de programación tales como variables, constantes y operadores en procesos de entrada-salida, así como en expresiones para resolver problemas. Utiliza un lenguaje de programación para codificar las soluciones de los problemas. Identifica y organiza los elementos que se requiere para resolver un problema. Elabora y evalúa expresiones matemáticas simples, aplicando la correcta prioridad de los operadores. Presenta sus productos en tiempo y forma, de tal manera que demuestra interés y cuidado en su trabajo. Fortalece el trabajo colaborativo y en equipo.	Historia del lenguaje C Entorno del IDE Elementos del lenguaje C Identificador Tipos de datos primitivos Variables Constantes Operadores y expresiones Palabras reservadas Comentarios	**5%**
2. Archivos .c de cada una de las estructuras de control, aplicando el uso de banderas, contadores y acumuladores.	Clasifica y aplica las diferentes estructuras de control en los problemas a resolver. Usa de manera correcta contadores, acumuladores y banderas.	Estructuras selectivas: Simple (if) Doble (if - else) Múltiple (switch)	**20%**

Utilizando buenas prácticas de programación como: ● Inicializar siempre las variables cuando se declaran. ● Expresar valores literales como constantes. ● Uso de sangría adecuado. ● Usar siempre llaves ({}) en las estructuras de control. ● Colocar al lado de una llave que cierre un bloque de código un indicativo de que tipo de estructura cierra.	Usa de manera correcta los operadores lógicos y relacionales en las expresiones (condiciones). Fortalece la abstracción en la solución de problemas. Mejora sus habilidades de comunicación oral y escrita.	Estructuras repetitivas: Contadores Acumuladores Banderas Mientras (while) Hacer mientras (do - while) Desde (for) Estructuras anidadas	
3. Archivos .c por cada tipo de dato presentar un programa tanto en vectores como matrices (al menos 6 programas). Utilizando buenas prácticas de programación como: ● Inicializar siempre las variables cuando se declaran. ● Expresar valores literales como constantes. ● Uso de sangría adecuado. ● Usar siempre llaves ({}) en las estructuras de control. ● Colocar al lado de una llave que cierre un bloque de código un indicativo de que tipo de estructura cierra.	Clasifica y aplica de manera adecuada los diferentes tipos de arreglos en los problemas a resolver. Presenta sus productos en tiempo y forma, de tal manera que demuestra interés y cuidado en su trabajo Refuerza la práctica del uso de las estructuras selectivas y repetitivas. Aplica operaciones sobre conjuntos de datos: ordena, analiza, promedia, entre otros. Aplica conocimientos en la práctica. Practica la toma de decisiones.	Definición de arreglo. Tipos de arreglos: Vectores Matrices	**10%**

4. Codificación de programas modulares que resuelvan problemas sencillos y que involucren el uso de los distintos tipos de funciones. Archivos .c por cada tipo de dato presentar un programa tanto en vectores como matrices (al menos 6 programas). Utilizando buenas prácticas de programación como: ● Inicializar siempre las variables cuando se declaran. ● Expresar valores literales como constantes. ● Usar de manera adecuada las variables globales y locales. ● Evitar el uso de sentencias goto, break. ● Usar un único return por función, que se colocará como última sentencia de la función. ● Evitar escribir funciones y procedimientos demasiado largos. ● Evitar copiar y pegar segmentos casi idénticos	Analiza y diseña una solución descendente de un problema dado. Descompone el problema en subproblemas. Codifica los subproblemas como funciones. Decide qué tipo de función implementa de acuerdo a cada subproblema. Usa adecuadamente variables locales y globales en la solución de sus problemas. Presenta sus productos en tiempo y forma, de tal manera que demuestra interés y cuidado en su trabajo	Definición de función Tipos de funciones: Sin paso de parámetros Con paso de parámetros: *Datos primitivos *Arreglo Funciones predefinidas: Matemáticas De manejo de caracteres	**10 %**

de código un mismo programa. ● Usar siempre llaves ({}) en las estructuras de control. ● Colocar al lado de una llave que cierre un bloque de código un indicativo de que tipo de estructura cierra.			
5. Codificación de programas modulares que resuelvan problemas sencillos y que involucren el uso de registros de datos. Archivos .c con registros (al menos 1 programa). Utilizando buenas prácticas de programación como: ● Inicializar siempre las variables cuando se declaran. ● Expresar valores literales como constantes. ● Usar de manera adecuada las variables globales y locales. ● Evitar el uso de sentencias goto, break. ● Usar un único return por función, que se colocará como última sentencia de la función.	Abstrae el problema organizando la información en tipos de datos compuestos complejos. Presenta sus productos en tiempo y forma, de tal manera que demuestra interés y cuidado en su trabajo Refuerza la práctica del uso de las estructuras selectivas y repetitivas. Aplica operaciones sobre conjuntos de datos: ordena, analiza, promedia, entre otros. Aplica conocimientos en la práctica. Practica la toma de decisiones.	Definición de registro Operaciones con registros: Entrada y salida de datos en registros. Arreglos de registros: Entrada, salida y manejo de datos en arreglos de registros.	**10%**

● Evitar escribir funciones y procedimientos demasiado largos. ● Evitar copiar y pegar segmentos casi idénticos de código un mismo programa. ● Usar siempre llaves ({}) en las estructuras de control. ● Colocar al lado de una llave que cierre un bloque de código un indicativo de que tipo de estructura cierra.			

Producto final		
Descripción	**Evaluación**	
Título: Análisis, diseño y codificación de un sistema de cómputo.	**Criterios de fondo:** Sistema funcional en base a los requerimientos establecidos. **Criterios de forma:** Presenta en tiempo y forma el sistema. Aplica las buenas prácticas de programación.	**Ponderación**
Objetivo: Aplicar el paradigma de programación estructurada en un problema dado.		
Caracterización La solución al problema debe contener las siguientes características: ● Análisis: identificar los elementos del problema y sus relaciones. ● Diseño de la solución: mediante la utilización de diagramas y / o pseudocódigos. ● Implementación: codificar considerando buenas prácticas de programación. La solución debe ser modular e incluir estructuras selectivas, repetitivas, datos estructurados como arreglos y registros. ● Funcionalidad: el programa debe cumplir con las especificaciones establecidas.		**35%**

Otros criterios		
Criterio	**Descripción**	**Ponderación**
Exposición del proyecto.	Expone y justifica su propuesta de solución, discute con sus compañeros de grupo sobre las ventajas y desventajas de la solución con respecto a las demás presentadas en el grupo	10%

6. REFERENCIAS Y APOYOS				
Referencias bibliográficas				
Referencias básicas				
Autor (Apellido, Nombre)	**Año**	**Título**	**Editorial**	**Enlace o biblioteca virtual donde esté disponible (en su caso)**
Joyanes Aguilar, L	2020	Fundamentos de programación	McGraw Hill. 5ta. Edición	
Referencias complementarias				
Márquez G., Osorio S., Olvera N	2011	Introducción a la Programación Estructurada en C	Pearson	
Juganaru Mathieu, M	2014	Introducción a la Programación	Grupo Editorial Patria	http://www.editorialpatria.com.mx/pdffiles/9786074384154.pdf

Apoyos (videos, presentaciones, bibliografía recomendada para el estudiante)

Unidad temática 1:

Pascual, F. L., López, E. P., & Ortim, F. O. (2000). CAPÍTULO 3: LENGUAJE ALGORÍTMICO. CONCEPTO DE PROGRAMA. (Spanish). Introducción a La Programación, Algoritmos y C/C++, 47–91.

Pascual, F. L., López, E. P., & Ortim, F. O. (2000). CAPÍTULO 2: TIPOS DE DATOS ELEMENTALES. (Spanish). Introducción a La Programación, Algoritmos y C/C++, 29–46.

Pascual, F. L., López, E. P., & Ortim, F. O. (2000). CAPÍTULO 4: INTRODUCCIÓN AL LENGUAJE C++. (Spanish). Introducción a La Programación, Algoritmos y C/C++, 93–112.

Pascual, F. L., López, E. P., & Ortim, F. O. (2000). CAPÍTULO 5: FUNCIONES BÁSICAS DE ENTRADA-SALIDA ENC++. (Spanish). Introducción a La Programación, Algoritmos y C/C++, 113–123.

Unidad temática 2:

https://elibro-net.wdg.biblio.udg.mx:8443/es/ereader/udg/74076?page=8

Pascual, F. L., López, E. P., & Ortim, F. O. (2000). CAPÍTULO 9: ESTRUCTURAS DE DATOS II: CADENAS DE CARACTERES. (Spanish). Introducción a La Programación, Algoritmos y C/C++, 238–256.

Unidad temática 3:
https://elibro-net.wdg.biblio.udg.mx:8443/es/ereader/udg/74076?page=8
Pascual, F. L., López, E. P., & Ortim, F. O. (2000). CAPÍTULO 8: ESTRUCTURAS DE DATOS I: VECTORES Y MATRICES. (Spanish). Introducción a La Programación, Algoritmos y C/C++, 207–236.

Unidad temática 4:
https://elibro-net.wdg.biblio.udg.mx:8443/es/ereader/udg/74076?page=8
Pascual, F. L., López, E. P., & Ortim, F. O. (2000). CAPÍTULO 11: ESTRUCTURAS DE DATOS IV: FICHEROS. (Spanish). Introducción a La Programación, Algoritmos y C/C++, 279–319.

Unidad temática 5:
https://elibro-net.wdg.biblio.udg.mx:8443/es/ereader/udg/74076?page=8

Pascual, F. L., López, E. P., & Ortim, F. O. (2000). CAPÍTULO 12: ESTRUCTURAS DE DATOS V: ESTRUCTURAS DINÁMICAS. (Spanish). Introducción a La Programación, Algoritmos y C/C++, 321–371.