Programa de asignatura
por competencias de educación superior

Sección I. Identificación del Curso

Tabla 1. Identificación de la Planificación del Curso.

Actualización:	Junio 01, 2022				
Carrera:	Ingeniería en Desarrollo de Software		**Asignatura:**	Programación orientada a objetos	
Academia:	Ciencias Computacionales y Programación /		**Clave:**	19SDS06	
Módulo formativo:	Programación aplicada		**Seriación:**	19SDS09 - Programación orientada a eventos	
Tipo de curso:	Presencial		**Prerrequisito:**	19SDS02 - Programación estructurada	
Semestre:	Segundo	**Créditos:** 10.12	**Horas semestre:**	162 horas	
Teoría:	2 horas	**Práctica:** 4 horas	**Trabajo indpt.:**	3 horas	**Total x semana:** 9 horas

Sección II. Objetivos educacionales

Tabla 2. Objetivos educacionales

Objetivos educacionales		Criterios de desempeño	Indicadores
OE1	Los egresados gestionarán recursos relacionados con el desarrollo de software en alguna organización.	Los egresados podrán aplicar metodologías en el desarrollo de proyectos en el contexto laboral.	20% de los egresados aplicarán metodologías en el desarrollo de software en su contexto laboral.
OE2	Los egresados diseñarán e implementarán soluciones innovadoras mediante el uso de tecnologías de la información.	Los egresados participarán activamente en el ciclo de desarrollo e integración continuos	25% de los egresados desempeñarán labores de desarrollo e integración continuos.
OE5	Los egresados serán capaces de emprender un negocio basado en el desarrollo de un producto o servicio de tecnologías de la información, aportando valor a la generación de empleos e incrementar el bienestar económico y social, de forma ecológica y sustentable.	Los egresados serán capaces de emprender un negocio basado en el desarrollo propio de un producto o servicio de tecnologías de la información.	2% de los egresados tendrán participación en el acta constitutiva de una empresa creada a partir del desarrollo de software para ofrecer un producto o servicio.
Atributos de egreso de plan de estudios		**Criterios de desempeño**	**Componentes**
AE1	Aplicar los conocimientos de ciencias básicas como física y matemáticas, así como las ciencias de la ingeniería para generar nuevos productos o servicios basándose en la innovación tecnológica.	Conocerá la terminología y la utilización de los punteros para el diseño de un programa orientado a objetos. Reconocerá el funcionamiento y la sintaxis de las clases y los objetos para el diseño de programas orientados a objetos, así como la implementación de las diferentes características que tiene esta programación.	1.1 Terminología básica. 1.2 Direcciones y punteros. 1.3 Declaración de punteros. 1.4 Operaciones con punteros. 1.4.1 Dirección. 1.4.2 Indirección. 1.5 Punteros a arreglos. 1.6 Punteros a constantes. 1.7 Funciones con parámetros por referencia. 1.7.1 Parámetros de salida. 1.7.2 Paso de arreglos por parámetros. 1.7.3 Paso de cadenas por parámetros.

Continuación: Tabla 2. Objetivos educacionales (continuación			
No.	Atributos de egreso de plan de estudios	Criterios de desempeño	Componentes
			2.1 Introducción al paradigma orientado a objetos. 2.2 Comparación de la POO con la programación estructurada. 2.3 Mecanismos básicos de la POO. 2.3.1 Abstracción. 2.3.2 Encapsulamiento. 2.3.3 Ocultamiento de información. 2.3.4 Clasificación. 2.3.5 Herencia. 2.3.6 Polimorfismo. 2.4 Clases. 2.4.1 Definición de una clase. 2.4.2 Modificadores de acceso a miembros. 2.4.3 Diagramas de clases. 2.5 El puntero this. 2.6 Constructor y Destructor. 2.7 Tipos de constructores. 2.7.1 Defecto. 2.7.2 Copia. 2.7.3 Inicialización. 2.8 Interacción entre objetos. 2.8.1 Diagramas de clases. 2.8.2 Composición. 2.8.3 Agregación. 2.9 Amistad. 2.9.1 Clases amigas. 2.9.2 Funciones amigas a un clase. 2.10 Herencia.

Continuación: Tabla 2. Objetivos educacionales (continuación			
No.	Atributos de egreso de plan de estudios	Criterios de desempeño	Componentes
			2.10.1 Clase base. 2.10.2 Clase derivada. 2.10.3 Reescritura de métodos. 2.10.4 Herencia como medio de especialización. 2.10.5 Herencia como medio de generalización. 2.10.6 Construcción y destrucción de clases heredadas. 2.11 Polimorfismo. 2.11.1 Métodos polimórficos. 2.11.2 Métodos abstractos. 2.11.3 Clases abstractas. 2.11.4 Clases abstractas puras. 2.12 Punteros a objetos que usan herencia. 2.12.1 Punteros a métodos no polimórficos. 2.12.2 Punteros a métodos polimórficos. 2.13 Tipos de datos abstractos. 2.14 Diferencia entre estructuras y clases en C++ 2.15 Patrones de diseño.
AE3	Desarrollar una experimentación adecuada para recopilar, almacenar y analizar grandes cantidades de información basándose en el juicio ingenieril para crear productos o servicios innovadores mediados por software.	Reconocerá el funcionamiento de la asignación dinámica de memoria en la programación orientada a objetos. Reconocerá el funcionamiento y las operaciones que tienen las estructuras de datos tipo lista, pila y cola, utilizando el paradigma de la programación orientada a objetos.	3.1 Traza de asignación dinámica de memoria. 3.2 Operadores new y delete. 3.3 Violaciones de acceso. 3.4 Fugas de memoria. 4.1 Listas Enlazadas.

Continuación: Tabla 2. Objetivos educacionales (continuación			
No.	Atributos de egreso de plan de estudios	Criterios de desempeño	Componentes
			4.1.1 Simples. 4.1.2 Dobles. 4.1.3 Circulares. 4.2 Pila y Cola. 4.2.1 Descripción de comportamiento. 4.2.2 Operaciones básicas. 4.2.3 Implementación mediante un adaptador de lista doble. 4.3 Árboles binarios de búsqueda. 4.3.1 Inserción. 4.3.2 Eliminación. 4.3.3 Búsqueda. 4.3.4 Recorridos. 4.4 Serialización. 4.4.1 Serialización de objetos. 4.4.2 Serialización de contenedores mediante campo longitud. 4.4.3 Serialización de contenedores mediante campo delimitador. 4.4.4 Serialización de árboles.

Sección III. Atributos de la asignatura

Tabla 3. Atributos de la asignatura

Problema a resolver		
Iniciar a los estudiantes con el conocimiento del paradigma de la programación orientada a objetos, así como la introducción de las estructuras dinámicas de datos con modelo orientado a objetos.		
Atributos (competencia específica) de la asignatura		
Identificar e implementar programas con el modelo o paradigma orientado a objetos, así como en la implementación de las estructuras de datos orientadas a objetos para la solución de problemáticas.		
Aportación a la competencia específica		**Aportación a las competencias transversales**
Saber	Saber hacer	Saber Ser
Conocer y analizar las características de la programación orientada a objetos, así como la importancia del manejo de este paradigma en los sistemas computacionales de la actualidad.	- Resolver problemas aplicando la metodología orientada a objetos en el desarrollo de programa en C++. - Identificar, plantear y resolver problemas específicos acordes a la Programación Orientada a Objetos y las Estructuras de Datos.	Trabajar en forma autónoma en el desarrollo de programas.
Producto integrador de la asignatura, considerando los avances por unidad		
Proyecto integrador, a partir de la creación de un programa, dando solución a una necesidad real en una organización, incorporando las competencias desarrolladas en cada una de las unidades de aprendizaje.		

Sección IV. Desglose específico por cada unidad formativa

Tabla 4.1. Desglose específico de la unidad "Manejo del tipo de dato Puntero."

Número y nombre de la unidad:	1. Manejo del tipo de dato Puntero.					
Tiempo y porcentaje para esta unidad:	Teoría:	4 horas	Práctica:	8 horas	Porcentaje del programa:	11.11%
Aprendizajes esperados:	Conocer la terminología y la utilización de los punteros para el diseño de un programa en C++.					

Temas y subtemas (secuencia)	**Criterios de desempeño**	**Estrategias didácticas**	**Estrategias de evaluación**	**Producto Integrador de la unidad (Evidencia de aprendizaje de la unidad)**
1.1 Terminología básica. 1.2 Direcciones y punteros. 1.3 Declaración de punteros. 1.4 Operaciones con punteros. 1.4.1 Dirección. 1.4.2 Indirección. 1.5 Punteros a arreglos. 1.6 Punteros a constantes. 1.7 Funciones con parámetros por referencia. 1.7.1 Parámetros de salida. 1.7.2 Paso de arreglos por parámetros. 1.7.3 Paso de cadenas por parámetros.	Saber: - Analizar y comprender la importancia de los punteros y cómo se implementan en un problema computacional. Saber hacer: - Realizar actividades relacionadas con lospunteros cumpliendo con las indicacionesestablecidas por parte del profesor y las envía por plataforma o la clase.	- Exposición por parte del profesor mediante algún material audiovisual. - Informe de lectura mediante una línea de tiempo. - Desarrollo de prácticas acorde al tema de punteros.	Evaluación formativa: - Ejercicios prácticos. Instrumento de evaluación: - Rúbrica para evaluar la calidad de los ejercicios prácticos. Evaluación sumativa: - Actividad integradora. Instrumento de evaluación: - Rúbrica para evaluar la actividad integradora de la unidad.	Planteamiento de un problema laboral o cotidiano en el que se pueda aplicar los punteros que represente la solución en un programa acorde a la estructura básica en C++.

Continuación: Tabla 4.1. Desglose específico de la unidad "Manejo del tipo de dato Puntero."				
Temas y subtemas (secuencia)	Criterios de desempeño	Estrategias didácticas	Estrategias de evaluación	Producto Integrador de la unidad
	Ser: Entregar los ejercicios propios (de su autoría) en tiempo y forma.			

Bibliografía

- Users Staff (2014). C++ Programación orientada a objetivos. 4° Edición. España: Creative Andina.
- Moreno, J.C. (2014). Programación orientada a objetos. 1° Edición. México: Ra-Ma.
- López, J.L. (2014). Programación orientada a objetos con C++ y Java. 1° Edición. México: Patria.

Sección IV. Desglose específico por cada unidad formativa

Tabla 4.2. Desglose específico de la unidad "Programación orientada a objetos."

Número y nombre de la unidad:	2. Programación orientada a objetos.					
Tiempo y porcentaje para esta unidad:	Teoría:	24 horas	Práctica:	48 horas	Porcentaje del programa:	66.67%
Aprendizajes esperados:	Reconocer el funcionamiento y la sintaxis de las clases y los objetos para el diseño de programas en C++, así como la implementación de las diferentes características que tiene la programación orientada a objetos.					

Temas y subtemas (secuencia)	**Criterios de desempeño**	**Estrategias didácticas**	**Estrategias de evaluación**	**Producto Integrador de la unidad (Evidencia de aprendizaje de la unidad)**
2.1 Introducción al paradigma orientado a objetos. 2.2 Comparación de la POO con la programación estructurada. 2.3 Mecanismos básicos de la POO. 2.3.1 Abstracción. 2.3.2 Encapsulamiento. 2.3.3 Ocultamiento de información. 2.3.4 Clasificación. 2.3.5 Herencia. 2.3.6 Polimorfismo. 2.4 Clases. 2.4.1 Definición de una clase. 2.4.2 Modificadores de acceso a miembros. 2.4.3 Diagramas de clases. 2.5 El puntero this. 2.6 Constructor y Destructor. 2.7 Tipos de constructores. 2.7.1 Defecto.	Saber: - Analizar y comprender la importancia del manejo y utilización de los punteros, así como su implementación en funciones conparámetros por referencia. Saber hacer: - Realizar ejercicios programados para lasactividades de cada subtema. - Realizar tareas de ejercicios de programas. Ser: Entregar los ejercicios propios (de su autoría)	- Usar las presentaciones con los temas descritos. - Complementar información con material audiovisual. - Resolver el problemario de ejercicios para programar.	Evaluación formativa: - Ejercicios prácticos. Instrumento de evaluación: - Rúbrica para evaluar la calidad de los ejercicios prácticos. Evaluación sumativa: - Proyecto final. Instrumento de evaluación: - Rúbrica para evaluar el proyecto final de la unidad.	Planteamiento de un problema laboral o cotidiano en el que se pueda aplicar la programación orientada a objetos y que represente la solución en unprograma acorde a este paradigma en C++.

Continuación: Tabla 4.2. Desglose específico de la unidad "Programación orientada a objetos."				
Temas y subtemas (secuencia)	Criterios de desempeño	Estrategias didácticas	Estrategias de evaluación	Producto Integrador de la unidad
2.7.2 Copia. 2.7.3 Inicialización. 2.8 Interacción entre objetos. 2.8.1 Diagramas de clases. 2.8.2 Composición. 2.8.3 Agregación. 2.9 Amistad. 2.9.1 Clases amigas. 2.9.2 Funciones amigas a un clase. 2.10 Herencia. 2.10.1 Clase base. 2.10.2 Clase derivada. 2.10.3 Reescritura de métodos. 2.10.4 Herencia como medio de especialización. 2.10.5 Herencia como medio de generalización. 2.10.6 Construcción y destrucción de clases heredadas. 2.11 Polimorfismo. 2.11.1 Métodos polimórficos. 2.11.2 Métodos abstractos. 2.11.3 Clases abstractas. 2.11.4 Clases abstractas puras. 2.12 Punteros a objetos que usan herencia. 2.12.1 Punteros a métodos no polimórficos. 2.12.2 Punteros a métodos polimórficos.	en tiempo y forma.			

Continuación: Tabla 4.2. Desglose específico de la unidad "Programación orientada a objetos."				
Temas y subtemas (secuencia)	Criterios de desempeño	Estrategias didácticas	Estrategias de evaluación	Producto Integrador de la unidad
2.13 Tipos de datos abstractos. 2.14 Diferencia entre estructuras y clases en C++ 2.15 Patrones de diseño.				

Bibliografía

- Users Staff (2014). C++ Programación orientada a objetivos. 4° Edición. España: Creative Andina.
- Moreno, J.C. (2014). Programación orientada a objetos. 1° Edición. México: Ra-Ma.
- López, J.L. (2014). Programación orientada a objetos con C++ y Java. 1° Edición. México: Patria.

Sección IV. Desglose específico por cada unidad formativa

Tabla 4.3. Desglose específico de la unidad "Asignación dinámica de memoria."

Número y nombre de la unidad:	3. Asignación dinámica de memoria.					
Tiempo y porcentaje para esta unidad:	Teoría:	2 horas	Práctica:	4 horas	Porcentaje del programa:	5.56%
Aprendizajes esperados:	Analizar el funcionamiento de la asignación dinámica de memoria en la programación para evitar posibles violaciones de acceso y fugas de memoria.					

Temas y subtemas (secuencia)	**Criterios de desempeño**	**Estrategias didácticas**	**Estrategias de evaluación**	**Producto Integrador de la unidad (Evidencia de aprendizaje de la unidad)**
3.1 Traza de asignación dinámica de memoria. 3.2 Operadores new y delete. 3.3 Violaciónes de acceso. 3.4 Fugas de memoria.	Saber: - Analizar los conceptos de la asignación dinámica de memoria y los operadores que son utilizados para evitar violaciones de acceso y fugas de memoria que se pudieran dar en un programa. Saber hacer: - Realizar ejercicios programados para las actividades de cada operador el new y eldelete. - Realizar tareas de ejercicios de programas.	- Usar las presentaciones con los temas descritos - Complementar información con material audiovisual. - Resolver el problemario de ejercicios para programar.	Evaluación formativa: - Ejercicios prácticos. Instrumento de evaluación: - Rúbrica para evaluar la calidad de los ejercicios prácticos. Evaluación sumativa: - Proyecto final. Instrumento de evaluación: - Rúbrica para evaluar la calidad del proyecto final.	Planteamiento de un problema laboral o cotidiano en el que se pueda aplicar la asignación dinámica de memoria que represente la solución en un programa acorde a este tema en C++.

Continuación: Tabla 4.3. Desglose específico de la unidad "Asignación dinámica de memoria."				
Temas y subtemas (secuencia)	Criterios de desempeño	Estrategias didácticas	Estrategias de evaluación	Producto Integrador de la unidad
	Ser: Entregar los ejercicios propios (de su autoría) en tiempo y forma.			

Bibliografía

- Users Staff (2014). C++ Programación orientada a objetivos. 4° Edición. España: Creative Andina.
- Moreno, J.C. (2014). Programación orientada a objetos. 1° Edición. México: Ra-Ma.
- López, J.L. (2014). Programación orientada a objetos con C++ y Java. 1° Edición. México: Patria.

Sección IV. Desglose específico por cada unidad formativa

Tabla 4.4. Desglose específico de la unidad "Estructuras dinámicas de datos."

Número y nombre de la unidad:	4. Estructuras dinámicas de datos.					
Tiempo y porcentaje para esta unidad:	Teoría:	6 horas	Práctica:	12 horas	Porcentaje del programa:	16.67%
Aprendizajes esperados:	Reconocer el funcionamiento y las operaciones que tienen las estructuras de datos tipo lista, pila y cola, utilizando el paradigma de la programación orientada a objetos para la resolución de problemas computacionales.					

Temas y subtemas (secuencia)	**Criterios de desempeño**	**Estrategias didácticas**	**Estrategias de evaluación**	**Producto Integrador de la unidad (Evidencia de aprendizaje de la unidad)**
4.1 Listas Enlazadas. 4.1.1 Simples. 4.1.2 Dobles. 4.1.3 Circulares. 4.2 Pila y Cola. 4.2.1 Descripción de comportamiento. 4.2.2 Operaciones básicas. 4.2.3 Implementación mediante un adaptador de lista doble. 4.3 Árboles binarios de búsqueda. 4.3.1 Inserción. 4.3.2 Eliminación. 4.3.3 Búsqueda. 4.3.4 Recorridos. 4.4 Serialización. 4.4.1 Serialización de objetos. 4.4.2 Serialización de contenedores mediante campo longitud. 4.4.3 Serialización de contenedores mediante campo delimitador.	Saber: - Conocer e identificar las estructuras dinámicas de datos tales como Listas,Pilas, Colas y Árboles para la resolución deproblemas computacionales. Saber hacer: - Realizar ejercicios programados para las actividades de cada estructura de datos. Realizar tareas de ejercicios de programas. Ser: Entregar los ejercicios propios (de su autoría) en tiempo y forma.	- Usar las presentaciones con los temas descritos. - Complementar información con material audiovisual. - Resolver el problemario de ejercicios para programar	Evaluación formativa: - Ejercicios prácticos. Instrumento de evaluación: - Rúbrica para evaluar la calidad de los ejercicios prácticos. Evaluación sumativa: - Proyecto final. Instrumento de evaluación: - Rúbrica para evaluar el proyecto final.	Planteamiento de un problema laboral o cotidiano en el que se pueda aplicar las estructuras de datos con los paradigmas orientados a objetos que represente la solución en un programa con mayor complejidad en C++.

Continuación: Tabla 4.4. Desglose específico de la unidad "Estructuras dinámicas de datos."				
Temas y subtemas (secuencia)	Criterios de desempeño	Estrategias didácticas	Estrategias de evaluación	Producto Integrador de la unidad
4.4.4 Serialización de árboles.				

Bibliografía

- Users Staff (2014). C++ Programación orientada a objetivos. 4° Edición. España: Creative Andina.
- Moreno, J.C. (2014). Programación orientada a objetos. 1° Edición. México: Ra-Ma.
- López, J.L. (2014). Programación orientada a objetos con C++ y Java. 1° Edición. México: Patria.

V. Perfil docente

Tabla 5. Descripción del perfil docente

Perfil deseable docente para impartir la asignatura
Carrera(s): Licenciatura o ingeniería en: -Informática. -Ciencias computacionales. -Ciencias de la informática. -Computación. -Computación e informática. -Desarrollo de aplicaciones computacionales. -Diseñador de programas de computación. -Informática administrativa. -Sistemas computacionales. -Cibernética y sistemas computacionales. -Sistemas computacionales e informáticos. -Ingeniero en Sistemas, titulado o carrera afín. o carrera afín - Con experiencia docente o en el campo deseable de 2 años. Manejo de TIC´s. Con habilidades pedagógicas y uso de metodologías alternativas de enseñanza.

- Experiencia mínima de dos años

- Licenciatura o superior