1. Datos Generales de la asignatura

Nombre de la asignatura:	Fundamentos de programación
Clave de la asignatura:	CDB-2408
SATCA[1]:	1-4-5
Carrera:	Ingeniería en Ciencia de Datos

2. Presentación

Caracterización de la asignatura

La asignatura contribuye significativamente al perfil de egreso al proporcionar a los estudiantes las habilidades y competencias necesarias para desarrollar programas de calidad. Al dominar los fundamentos de la programación, los estudiantes adquieren la capacidad de diseñar algoritmos eficientes, implementar soluciones efectivas a problemas computacionales y trabajar de manera colaborativa en proyectos de desarrollo de software.

La asignatura es una habilidad fundamental en la era digital actual. Desde el desarrollo de aplicaciones móviles hasta el análisis de datos y la automatización de procesos, la programación está en el corazón de la innovación tecnológica. Los conocimientos adquiridos en esta asignatura sientan las bases para el éxito en campos como la Ciencia de Datos, Ingeniería de Software, inteligencia artificial entre otras.

La asignatura introduce a los estudiantes en los conceptos básicos de la programación, incluyendo tipos de datos, estructuras de control (como bucles y condicionales), funciones, estructuras de datos elementales (como arreglos).

La asignatura se relaciona estrechamente con asignaturas como programación Orientada a Objetos, Programación Avanzada para Ciencia de Datos, Machine Learning e Ingeniería de Software. En términos de competencias específicas, la asignatura contribuye al desarrollo de habilidades como el pensamiento computacional, la resolución de problemas, la abstracción, el diseño algorítmico, la depuración de código, la colaboración en equipos de desarrollo y la capacidad de comunicar ideas técnicas de manera efectiva.

Intención didáctica

Los contenidos didácticos deben ser presentados de manera gradual y estructurada, comenzando por conceptos básicos y avanzando hacia temas más complejos. La asignatura se organiza en cinco temas donde se aborda el diseño de algoritmos, introducción a la programación, control de flujo, arreglos y funciones.

[1] Sistema de Asignación y Transferencia de Créditos Académicos

El enfoque con que se deben abordar los temas debe ser principalmente práctico, con énfasis en la resolución de problemas y la escritura de código. Se debe fomentar la comprensión de los conceptos teóricos a través de la práctica activa, utilizando ejemplos concretos y situaciones del mundo real siempre que sea posible.

Los contenidos deben ser presentados de manera que los estudiantes adquieran una comprensión sólida de los fundamentos de la programación. Se debe cubrir una amplia gama de temas, desde conceptos básicos como variables y estructuras de control hasta técnicas más avanzadas como la programación modular (funciones) y arreglos.

Se deben destacar actividades que fomenten el trabajo en equipo, la resolución de problemas, la comunicación efectiva y la creatividad. Los estudiantes deben participar en la resolución de ejercicios prácticos, proyectos de programación colaborativos y presentaciones orales para desarrollar estas competencias genéricas.

Las competencias genéricas que se desarrollan incluyen habilidades de comunicación, trabajo en equipo, pensamiento crítico, resolución de problemas, autogestión del aprendizaje y creatividad. Estas competencias son fundamentales para el éxito tanto en la programación como en cualquier otro campo profesional.

El docente debe desempeñar un papel activo y facilitador en el proceso de enseñanza y aprendizaje. Debe proporcionar orientación y apoyo a los estudiantes, fomentar la participación activa en clase, brindar retroalimentación constructiva sobre el progreso de los estudiantes y crear un ambiente de aprendizaje inclusivo y colaborativo. Además, el docente debe estar actualizado en cuanto a las tendencias y avances en el campo de la programación para proporcionar a los estudiantes una educación relevante y de calidad.

3. Participantes en el diseño y seguimiento curricular del programa

Lugar y fecha de elaboración o revisión	Participantes	Observaciones
Instituto Tecnológico Superior de Alvarado del 21 al 23 agosto de 2023.	Representante del Instituto Tecnológico Superior de Alvarado.	Propuesta inicial.
Tecnológico Nacional de México 30 octubre 2023	Representante del Instituto Tecnológico de: Querétaro y del Instituto Tecnológico Superior de Alvarado.	Presentación de la propuesta de la carrera de Ingeniería en Ciencia de Datos.

Instituto Tecnológico de Querétaro Campus Norte del 19 al 22 de marzo 2024.	Representantes de los Institutos Tecnológicos de: Morelia, Puebla, Querétaro, Tehuacán. Instituto Tecnológico Superior de Alvarado. CENIDET. Representante de Ciencias Básica de los Institutos de: Celaya, Morelia y CIIDET.	Diseño y/o desarrollo curricular de la carrera de Ingeniería en Ciencia de Datos.
Tecnológico Nacional de México del 22 al 24 de abril del 2024	Representante del Instituto Tecnológico de Querétaro e Instituto Tecnológico Superior de Alvarado.	Contraste y ajuste de las asignaturas de Ingeniería en Ciencia de Datos con respecto a las de Ing. en Inteligencia Artificial, Ing. en Desarrollo WEB e Ing. en Ciberseguridad
Tecnológico Nacional de México del 27 al 31 de mayo del 2024.	Representantes de los Institutos Tecnológicos de: Morelia, Querétaro. Instituto Tecnológico Superior de Alvarado. CENIDET.	Consolidación curricular de la carrera de Ingeniería Ciencia de Datos

4. Competencia(s) a desarrollar

Competencia(s) específica(s) de la asignatura
Aplica algoritmos y lenguajes de programación para la implementación en el diseño de soluciones a problemáticas reales de la sociedad.

5. Competencias previas

Capacidad de análisis y pensamiento crítico.

6. Temario

No.	Temas	Subtemas
1	Diseño algorítmico	1.1. Definición de conceptos básicos. 1.2. Representación de algoritmos: diagramas de flujo y pseudocódigo. 1.3. Diseño de algoritmos: entender el problema, definición de objetivos, identificar las entradas y salidas, división del problema en pasos más pequeños (si es que aplica), selección de estructuras de control adecuadas, escribir el pseudocódigo o el diagrama de flujo, implementar y probar el algoritmo y documentar el algoritmo. 1.4. Técnicas de resolución de problemas: definir claramente el problema, divide y vencerás y análisis causa-efecto (diagrama Ishikawa).
2	Introducción a la Programación.	2.1. Definición de conceptos básicos. 2.2. Características del lenguaje de programación. 2.3. Estructura básica de un programa. 2.4. Elementos del lenguaje: tipos de datos, constantes, variables, comentarios, identificadores, operadores y entrada y salida de datos. 2.5. Entorno de desarrollo para el lenguaje de programación: compilación, enlace, ejecución y errores.
3	Control de Flujo	3.1. Estructuras secuenciales 3.2. Estructuras selectivas: simple, doble y múltiple. 3.3. Estructuras iterativas: repetir mientras, hasta, desde. 3.4. Estructuras anidadas: selectivas e iterativas.
4	Arreglos	4.1. Conceptos básicos de arreglos 4.2. Arreglos unidimensionales: operaciones y aplicaciones. 4.3. Arreglos multidimensionales: operaciones y aplicaciones.

5	Funciones	5.1. Conceptos básicos de funciones. 5.2. Definición, declaración y llamada de una función. 5.3. Paso de parámetros o argumentos. 5.4. Aplicación de funciones 5.5. Principio de Única Responsabilidad.

7. Actividades de aprendizaje de los temas

1. Diseño algorítmico	
Competencias	Actividades de aprendizaje
Específica(s): Aplica los conceptos básicos, herramientas para el diseño de algoritmos y técnicas de resolución de problemas. *Genéricas:* ● Habilidades de gestión de información (habilidad para buscar y analizar información proveniente de fuentes diversas). ● Capacidad de análisis y síntesis. ● Capacidad de comunicación oral y escrita. ● Capacidad de aplicar los conocimientos en la práctica. ● Habilidades en el uso de las tecnologías de la información y de la comunicación.	● Realizar un glosario de términos o un cuestionario sobre los conceptos básicos de programación como son: Algoritmo, Lenguaje de Programación, Variables, Tipos de Datos, Estructuras de Control, Funciones, Arreglos, Estructura de Datos, Modularidad, Comentarios, Recursión entre otros. ● Describir una problemática de la vida real y plantear una solución a través de un algoritmo. Representar dicha solución a través de un diagrama de flujo o pseudocódigo. ● Describir una problemática de la vida real y plantear una solución a través de la técnica divide y vencerás.
2. Introducción a la programación	
Competencias	Actividades de aprendizaje
Específica(s): Aplica un lenguaje de programación para la resolución de problemas. *Genéricas:* ● Habilidades de gestión de información (habilidad para buscar y analizar información proveniente de fuentes diversas). ● Capacidad de análisis y síntesis. ● Capacidad de Comunicación oral y escrita en su propia lengua. ● Capacidad de aplicar los conocimientos en la práctica.	● Asignar a los estudiantes problemas de codificación simples que requieran el uso de conceptos básicos de programación, como variables, expresiones, estructuras de control y funciones. ● Utilizar laboratorios virtuales o entornos de programación en línea donde los estudiantes puedan practicar escribir y ejecutar código en un entorno controlado. ● Configurar ejercicios guiados que los estudiantes puedan completar paso a paso para familiarizarse con la sintaxis y la estructura básica de un programa.

● Habilidades en el uso de las tecnologías de la información y de la comunicación.	● Dividir a los estudiantes en grupos y asignarles proyectos pequeños para implementar soluciones a problemas específicos. ● Plantear problemas del mundo real que puedan resolverse mediante programación y pedir a los estudiantes que diseñen y desarrollen soluciones utilizando los conceptos aprendidos. ● Proporcionar a los estudiantes código con errores y pedirles que identifiquen y corrijan los errores utilizando técnicas de depuración. ● Proporcionar retroalimentación constructiva sobre el trabajo realizado por los estudiantes, destacando tanto los aspectos positivos como las áreas de mejora. ● Evaluar el progreso de los estudiantes mediante pruebas de evaluación formativa, ejercicios prácticos y proyectos completados.
3. Control de flujo	
Competencias	Actividades de aprendizaje
Específica(s): Aplica las estructuras condicionales y repetitivas de un lenguaje de programación para resolución de problemas reales. *Genéricas:* ● Habilidades de gestión de información (habilidad para buscar y analizar información proveniente de fuentes diversas). ● Capacidad de análisis y síntesis. ● Capacidad de comunicación oral y escrita. ● Capacidad de aplicar los conocimientos en la práctica. ● Habilidades en el uso de las tecnologías de la información y de la comunicación.	● Proporcionar a los estudiantes una serie de problemas simples que requieran el uso de estructuras de control como condicionales (if-else) y bucles (for, while). Posterior a ello, solicitar a los estudiantes que implementen soluciones utilizando un lenguaje de programación específico. ● Utiliza herramientas en línea o software interactivo para simular la ejecución de código con diferentes estructuras de control. ● Proporcionar a los estudiantes fragmentos de código con estructuras de control y solicitar que analicen su comportamiento. Solicitar a los estudiantes que interpreten la salida esperada del programa y cómo se alteraría si se modifican las condiciones. ● Presentar a los estudiantes problemas del mundo real que requieran el uso de estructuras de control para su resolución. Por ejemplo, la gestión de inventario, la automatización de procesos, o la simulación de sistemas físicos.

4. Arreglos	
Competencias	Actividades de aprendizaje
Específica(s): Aplica estructuras de datos (arreglos) en un lenguaje de programación que permitan la organización de datos en la resolución de problemas reales. *Genéricas:* ● Habilidades de gestión de información (habilidad para buscar y analizar información proveniente de fuentes diversas). ● Capacidad de análisis y síntesis. ● Capacidad de comunicación oral y escrita. ● Capacidad de aplicar los conocimientos en la práctica. ● Habilidades en el uso de las tecnologías de la información y de la comunicación.	● Proporcionar a los estudiantes una serie de ejercicios que requieran acceder a elementos individuales de un arreglo y manipular su contenido. Solicitar a los estudiantes que realicen operaciones como la asignación de valores, la modificación de elementos y la impresión de elementos específicos. ● Asignar problemas que impliquen buscar un elemento específico en un arreglo o ordenar los elementos en orden ascendente o descendente. Solicitar a los estudiantes que implementen algoritmos de búsqueda (lineal o binaria) y algoritmos de ordenamiento (burbuja, selección, inserción, etc.).
5. Funciones	
Competencias	Actividades de aprendizaje
Específica(s): Aplicar el modularidad (funciones) en el desarrollo de programas para la optimización de los mismos y reutilización de código. *Genéricas:* ● Habilidades de gestión de información (habilidad para buscar y analizar información proveniente de fuentes diversas). ● Capacidad de análisis y síntesis. ● Capacidad de comunicación oral y escrita. ● Capacidad de aplicar los conocimientos en la práctica ● Habilidades en el uso de las tecnologías de la información y de la comunicación.	● Investigar las ventajas, desventajas, estructura y usos de la programación modular y presentar un reporte. ● Gestionar información sobre la declaración y el uso de métodos o funciones y presentarla en un resumen. ● Identificar la diferencia entre parámetros y argumentos, su estructura e importancia, mediante una investigación bibliográfica, presentando ejemplos de programas en exposición grupal. ● Realizar una práctica de ejercicios que involucren la implementación de métodos o funciones con pase de parámetros en la resolución de problemas del contexto, documentar y exponer.

8. Práctica(s)

- Elaborar diagrama de flujo, pseudocódigo y programa a problemas planteados que impliquen entrada y salida de datos, que impliquen declaración de variables y uso de expresiones aritméticas.
- Solucionar problemas planteados en la clase a través de diagramas de flujo y pseudocódigo y codificarlos para comprobar su funcionamiento.
- Resolver ejercicios que involucren diferentes expresiones algebraicas que ilustren la precedencia de los operadores aritméticos, utilizar un programa sugerido por el docente para comprobar los resultados obtenidos.
- En un lenguaje de programación, definir y manipular diferentes tipos de datos y comprobar sus características a través de un programa asignado por el docente.
- Elaborar ejercicios que impliquen estructuras secuenciales y selectivas entregando pseudocódigo, diagrama de flujo y programa.
- Elaborar ejercicios que impliquen las estructuras iterativas soportadas por un lenguaje de programación, entregando pseudocódigo, diagrama de flujo y programa.
- Elaborar ejercicios que impliquen organización de datos por medio de la implementación de arreglos entregando diagrama de flujo, pseudocódigo y programa.
- Elaborar ejercicios que impliquen organización de datos por medio de la implementación estructuras o registros entregando diagrama de flujo, pseudocódigo y programa.
- Elaborar ejercicios que impliquen la implementación de funciones entregando diagrama de flujo, pseudocódigo y programa.

9. Proyecto de asignatura

El objetivo del proyecto que planteé el docente que imparta esta asignatura, es demostrar el desarrollo y alcance del(los) logro(s) formativo(s) de la asignatura, considerando las siguientes fases:

- **Fundamentación:** marco referencial (teórico, conceptual, contextual, legal) en el cual se fundamenta el proyecto de acuerdo con un diagnóstico realizado, mismo que permite a los estudiantes lograr la comprensión de la realidad o situación objeto de estudio para definir un proceso de intervención o hacer el diseño de un modelo.
- **Planeación:** con base en el diagnóstico en esta fase se realiza el diseño del proyecto por parte de los estudiantes con asesoría del docente; implica planificar un proceso: de intervención empresarial, social o comunitario, el diseño de un modelo, entre otros, según el tipo de proyecto, las actividades a realizar los recursos requeridos y el cronograma de trabajo.
- **Ejecución:** consiste en el desarrollo de la planeación del proyecto realizada por parte de los estudiantes con asesoría del docente, es decir en la intervención (social, empresarial), o construcción del modelo propuesto según el tipo de proyecto, es la fase de mayor duración que implica el desempeño de los saberes, habilidades y destrezas a desarrollar.
- **Evaluación:** es la fase final que aplica un juicio de valor en el contexto laboral-profesión, social e investigativo, ésta se debe realizar a través del reconocimiento de logros y aspectos a mejorar se estará promoviendo el concepto de “evaluación para la mejora continua”, el desarrollo del pensamiento crítico y reflexivo en los estudiantes.

10. Evaluación por competencias

La evaluación debe hacerse diagnóstica, formativa y sumativa. De igual manera, para fortalecer la parte actitudinal, se recomienda guiar al estudiante hacia la introspección para utilizar la autoevaluación y la coevaluación.

En el caso de las actividades de aprendizaje se sugiere el uso de estrategias metacognitivas como: mapas conceptuales, reportes de prácticas, exposiciones en clase, ensayos, resúmenes, reportes de visitas industriales, trípticos, guías de entrevista, observación y cuestionarios. Mientras que para verificar el nivel del logro de las competencias del estudiante se recomienda utilizar: el portafolio de evidencias, listas de cotejo, rúbricas, matrices de valoración, guías de observación, además de estrategias en las que se logren las competencias blandas.

11. Fuentes de Información

1. Albert, R., y Breedlove, T. (2009) C++: An active learning approach. USA: Jones and Bartlett Publishers
2. Brassard G.& et all. (Sin fecha). Fundamentos de Algoritmia. Pearson: Prentice Hall
3. Cairo Battistutti, O., (2005), *Metodología de la Programación, Algoritmos Diagrama de Flujo y Programas*, Ciudad, Estado, Provincia, País: Alfaomega. ISBN 970-15-1100-X
4. Cheng, H. H. (2010). *C for engineers and scientists: An interpretive approach*. USA: McGraw-Hill Higher Education
5. Deitel, P y Deitel, H. (2008) *Como Programar en C++.* México:Pearson Prentice Hall.
6. Deitel, J., y Deitel, M. (2012) C++: How to program. USA: Prentice Hall.
7. Deitel, H. (2008). Java cómo programar. (7ª ed.) Prentice Hall México, 2008. ISBN 9789702611905
8. García Molina, J.J., (2005), *Introducción a la programación un Enfoque Algorítmico*, Ciudad, Estado, Provincia, País: Paraninfo.
9. Joyanes, L. (2012) Fundamentos generales de programación. España;McGraw Hill.
10. Joyanes, L. (2008) *Fundamentos de programación: algoritmos, estructura de datos y objetos*. España: McGraw-Hill.
11. Joyanes, L. (2010) *Programación en C, C++, Java y UML*. México: McGraw-Hill. ISBN 978-970-10-6949-3.
12. Joyanes, L., Fernández, M. y Rodríguez L. (2003) *Fundamentos de Programación Libro de Problemas Algoritmos Estructura de Datos y Objetos*. México:Mc. Graw Hill.
13. Kamthane, N. (2011) *Programming in C*. India: Dorling Kindersley.
14. López, L. (2011) Programación Estructurada y Orientada a Objetos México:Alfaomega.
15. Marquez, G. (2012) Introducción a la programación estructurada en C. España:Pearson
16. McMilan, M. (2011) Learning C++. USA: InfinateSkills.
17. Méndez, A. (2013) Diseño de algoritmos y su programación en C. México:Alfaomega.
18. Mothe, M. (2012) *C++ programming: a practical approach*. India: Dorling Kindersley.
19. Urrutia, G. (2012) Curso de Programación en C para principiantes. España:@Gorka Urrutia
20. Zavala, R y Llamas, R. (2013) Fundamentos de programación para principiantes. España:McGraw Hill.