1.- DATOS DE LA ASIGNATURA

Nombre de la asignatura:	**Fundamentos de programación**
Carrera:	**ISC**
Clave de la asignatura:	**SCD1008**
(Créditos) SATCA[1]	**2-3-5**

2.- PRESENTACIÓN

Caracterización de la asignatura.

Esta asignatura aporta al estudiante de la carrera de ISC, la capacidad para desarrollar un pensamiento lógico, identificar el proceso de creación de un programa y desarrollo de algoritmos para resolver problemas con un enfoque orientado a objetos.

La asignatura proporciona al estudiante una herramienta para resolver problemas de aplicaciones de la vida ordinaria y de aplicaciones de la ingeniería.

Está diseñada para el logro de competencias específicas dirigidas al aprendizaje de los dominios: manejo de consola y diseño de algoritmos. Comprenderá los conceptos básicos de la programación orientada a objetos y escribirá expresiones aritméticas y lógicas en un lenguaje de programación orientado a objetos. Así como el uso y funcionamiento de las estructuras secuenciales, selectivas, arreglos unidimensionales, bidimensionales y métodos en el desarrollo de aplicaciones. Será capaz de aplicarlos al construir y desarrollar aplicaciones de software que requieran dichas estructuras.

Este curso genera las competencias necesarias para que el alumno desarrolle aplicaciones que den solución a los problemas que le plantee la vida diaria.

[1] Sistema de asignación y transferencia de créditos académicos

Fundamentos de Programación es el soporte directo de las asignaturas: Programación Orientada a Objetos, Estructura de Datos, Tópicos Avanzados de Programación y de forma indirecta se relaciona con el desarrollo de sistemas de software, Sistemas Operativos y Programación de Sistemas.

Intención didáctica.

La asignatura proporciona al alumno los conceptos esenciales del diseño algorítmico, manteniendo este dentro del paradigma orientado a objetos pero sin profundizar en el mismo. Se organiza el temario en cinco unidades.

En la **primera unidad** se estudian los conceptos básicos para introducir al estudiante en la metodología para resolver problemas a través de la computadora y modelos conceptuales para abordar las siguientes unidades temáticas. Es importante concientizar al estudiante sobre la importancia del planteamiento de modelos metodológicos para la solución de problemas cotidianos y de ingeniería utilizando el razonamiento lógico.

La **segunda unidad**, tiene la finalidad de introducir al estudiante en el paradigma orientado a objetos para diseñar e implementar soluciones en un lenguaje de programación orientada objetos, utilizando los conceptos adquiridos.

En la **tercera unidad** el estudiante será capaz de conocer y dominar las características y el entorno de programación del lenguaje de programación orientada a objetos denominado C#.

La **cuarta unidad** tiene como objetivo que el estudiante identifique, comprenda, seleccione e implemente la estructura de control más adecuada a un problema específico, dado que es común encontrar en la práctica problemas cuyas operaciones están condicionadas o deban ejecutarse un número repetido de veces.

En la **quinta unidad** el estudiante será capaz de analizar, comprender, diseñar e implementar métodos como una herramienta de agrupación de bloques de sentencias de código de tal manera que organicen de forma más eficiente sus aplicaciones. Esta unidad es de suma

importancia para que el estudiante comprenda, diseñe, implemente y aplique las acciones realizadas por los objetos en la solución de problemas cotidianos y de ingeniería; también es fundamental para que el estudiante muestre un buen desempeño en las materias posteriores (Programación Orientada a Objetos, Estructuras de Datos, Graficación, Tópicos Avanzados de Programación y otras).

La **sexta unidad** tiene la finalidad de implementar arreglos para una gran variedad de propósitos que proporcionan un medio conveniente de agrupar variables relacionadas y organizar datos de una manera que puedan ser fácilmente procesados.

3.- COMPETENCIAS A DESARROLLAR

Competencias específicas:	**Competencias genéricas:**
Analizar, diseñar y desarrollar soluciones de problemas reales utilizando algoritmos computacionales para implementarlos en un lenguaje de programación orientado a objetos.	**Competencias instrumentales** • Capacidad de análisis y síntesis. • Capacidad de pensamiento lógico, algorítmico, analítico y sintético. • Resolución de problemas. • Toma de decisiones. • Destrezas tecnológicas relacionadas con el uso de maquinaria, destrezas de computación. • Búsqueda y manejo de información. **Competencias interpersonales** • Capacidad crítica y autocrítica • Habilidades interpersonales **Competencias sistémicas**

	• Capacidad de aplicar los conocimientos en la práctica • Habilidades de investigación • Capacidad de aprender • Capacidad de generar nuevas ideas (creatividad). • Habilidad para trabajar en forma autónoma. • Búsqueda del logro

4.- HISTORIA DEL PROGRAMA

Lugar y fecha de elaboración o revisión	**Participantes**	**Observaciones (cambios y justificación)**
Instituto Tecnológico de Nuevo Laredo. Fecha: 9 de junio del 2016.	Ing. Gloria María Rodríguez Morales, Ing. Verónica Dávila Toscano, Ing. Bertha Alicia Fernández De la Mora, Ing. Thelma Gpe. Cantú Treviño, Ing. Bruno López Takeyas, Ing. Jaime David Johnston Barrientos.	• **Se introduce el paradigma orientado a objetos para facilitar la programación de las aplicaciones y minimizar la dificultad de transición al curso de POO y posteriores** • **Se desarrollarán aplicaciones en modo consola utilizando el lenguaje C#**

5.- OBJETIVO(S) GENERAL(ES) DEL CURSO (competencias específicas a desarrollar en el curso)

Analizar, diseñar e implementar aplicaciones para solucionar problemas reales utilizando algoritmos computacionales utilizando un lenguaje de programación orientado a objetos.

6.- COMPETENCIAS PREVIAS

- Ninguna

7.- TEMARIO

Unidad	Temas	Subtemas
1	Conceptos básicos	1.1 Algoritmos 1.1.1 Concepto 1.1.2 Características 1.1.3 Representación 1.1.3.1 Pesudocódigo 1.1.3.2 Diagrama de flujo 1.2 Modelo de Von Neumann 1.2.1 Entrada 1.2.2 CPU 1.2.2.1 ALU 1.2.2.2 CU 1.2.2.3 Registros 1.2.3 Salida 1.2.4 Memoria principal (RAM) 1.2.5 Memoria secundaria 1.3 Metodología para resolver problemas a través de la computadora 1.3.1 Análisis 1.3.1.1 Investigación preliminar 1.3.1.2 Descripción del problema 1.3.1.3 Especificaciones de entrada 1.3.1.4 Especificaciones de salida 1.3.1.5 Variables auxiliares 1.3.2 Diseño 1.3.2.1 Diseño descendente (*top-down*) 1.3.2.2 Refinamiento por pasos 1.3.2.3 Herramientas de diseño 1.3.2.4 Pruebas de diseño 1.3.3 Implementación 1.3.3.1 Codificación 1.3.3.2 Ejecución 1.3.3.3 Comprobación 1.4 Datos y tipos de datos 1.4.1 Numéricos 1.4.1.1 Enteros

		1.4.1.2 Reales 1.4.2 Alfanuméricos 1.4.2.1 Caracter 1.4.2.2 Cadena 1.4.3 Lógicos o booleanos 1.4.4 Otros 1.5 Operadores aritméticos 1.5.1 Suma (+) 1.5.2 Resta (-) 1.5.3 Multiplicación (*) 1.5.4 División (/) 1.5.5 Entera 1.5.6 Real 1.5.7 Prototipar (*cast*) 1.6 Reglas de prioridad de los operadores aritméticos 1.7 Funciones matemáticas de la clase *Math* 1.7.1 Exponenciación (*Pow*) 1.7.2 Raíz cuadrada (*Sqrt*) 1.7.3 Trigonométricas (*Sin*, *Cos*, *Tan*) 1.8 Contadores y acumuladores 1.8.1 Incrementos (++) 1.8.2 Decrementos (--) 1.8.3 Acumuladores (+=, -=, *=, /*)
2	Clases y objetos	2.1 Introducción al paradigma orientado a objetos 2.2 Clases, objetos, abstracción y encapsulamiento 2.3 Modificadores de acceso 2.3.1 Público 2.3.2 Privado 2.4 Diagramas de clases en UML 2.5 Nomenclatura para identificar los componentes de un proyecto con clases 2.6 Declaración de clases, atributos y propiedades 2.7 Diagramas de clases en UML 2.8 Declaración de clases, atributos y propiedades

3	Introducción a la programación en C#	3.1 Características del lenguaje de programación 3.2 Estructura básica de un programa. 3.3 Traducción de un programa: Compilación y errores en tiempo de compilación. 3.4 Ejecución de un programa. 3.5 Elementos del lenguaje: datos, literales y constantes, identificadores, variables, parámetros, operadores, entrada y salida de datos. 3.6 Errores en tiempo de ejecución.
4	Control de flujo	4.1 Estructuras secuenciales. 4.1.1 Entrada y salida de datos 4.1.2 Manejo de excepciones 4.1.2.1 Sentencias *try*, *catch* y *finally* 4.1.2.2 Excepciones generadas por el programador (sentencia *throw*) 4.2 Estructuras selectivas 4.2.1 Simples (*if*) 4.2.2 Dobles (*if-then*) 4.2.3 Múltiples (*switch*) 4.3 Operadores relacionales 4.3.1 Menor que (<) 4.3.2 Mayor que (>) 4.3.3 Menor o igual que (<=) 4.3.4 Mayor o igual que (>=) 4.3.5 Igual que (==) 4.3.6 Diferente (!=) 4.4 Operadores lógicos 4.4.1 AND (&&) 4.4.2 OR (||) 4.4.3 NOT (!) 4.4.4 Tablas de verdad 4.4.5 Reglas de prioridad de los operadores lógicos 4.5 Estructuras iterativas 4.5.1 Repetir (*for*) 4.5.2 Mientras (*while*) 4.5.3 Hacer-mientras (*do-while*) 4.6 Estructuras iterativas anidadas

5	Métodos	5.1 Conceptos 5.2 Tipos de métodos 5.2.1. Procedimientos 5.2.2. Funciones 5.4.2.1. Recepción del valor devuelto 5.3 Envío de parámetros 5.3.1 Por valor 5.3.2 Por referencia (*ref*) 5.3.3 De salida (*out*) 5.4 Ámbito de las variables 5.4.1 Locales 5.4.2 Globales 5.5 Representación algorítmica de los métodos 5.5.1 En pseudocódigo 5.5.2 En diagrama de flujo
6	Arreglos	6.1 Unidimensionales 6.1.1 Conceptos básicos 6.1.2 Representación 6.1.3 Operaciones 6.1.4 Aplicaciones 6.2 Bidimensionales 6.2.1 Conceptos básicos 6.2.2 Representación 6.2.3 Operaciones 6.2.4 Aplicaciones

8.- SUGERENCIAS DIDÁCTICAS (desarrollo de competencias genéricas)

El profesor debe:

Ser conocedor de la disciplina que está bajo su responsabilidad. Desarrollar la capacidad para coordinar y orientar el trabajo del estudiante, potenciar en él la autonomía, el trabajo cooperativo y la toma de decisiones.

- Proponer problemas que:
  - Propicien el desarrollo de la lógica de programación.
  - Permitan al estudiante la integración de los contenidos, para su análisis y solución.
  - Fortalezcan la comprensión de conceptos que serán utilizados en materias posteriores.

- Proponer actividades de búsqueda, selección y análisis de información en distintas fuentes.
- Propiciar el uso de las nuevas tecnologías en el desarrollo de los contenidos de la asignatura.
- Propiciar, en el estudiante, el desarrollo de actividades intelectuales de inducción-deducción y análisis-síntesis, las cuales lo encaminan hacia la investigación, la aplicación de conocimientos y la solución de problemas.
- Desarrollar la capacidad de abstracción, análisis y síntesis.
- Fomentar el uso de las convenciones en la codificación de un algoritmo.
- Relacionar los contenidos de la asignatura con el respeto al marco legal, el cuidado del medio ambiente y con las prácticas de una ingeniería con enfoque sustentable.

9.- SUGERENCIAS DE EVALUACIÓN

La evaluación de la asignatura debe de ser continua y se debe considerar el desempeño en cada una de las actividades de aprendizaje, haciendo especial énfasis en obtener evidencias de aprendizaje como:

- Información obtenida durante las investigaciones solicitadas, plasmadas en documentos escritos o digitales
- Solución algorítmica a problemas reales o de ingeniería utilizando el diseño escrito o en herramientas digitales
- Codificación en un lenguaje de programación orientada a objeto de las soluciones diseñadas
- Participación y desempeño en el aula y laboratorio
- Dar seguimiento al desempeño en el desarrollo del temario (dominio de los conceptos, capacidad de la aplicación de los conocimientos en problemas reales y de ingeniería)
- Se recomienda utilizar varias técnicas de evaluación con un criterio específico para cada una de ellas (teórico-práctico).
- Desarrollo de un proyecto por unidad que integre los tópicos vistos en la misma
- Desarrollo de un proyecto final que integre todas las unidades de aprendizaje

- Uso de una plataforma educativa en internet la cual puede utilizarse como apoyo para crear el portafolio de evidencias del alumno (integrando: tareas, prácticas, evaluaciones, etc.)
- Se recomienda la evaluación las unidades mediante la revisión de programas de tarea, aplicación de un examen teórico, un examen práctico y la solución de un cuestionario en línea utilizando la plataforma del Synesi2.

10.- UNIDADES DE APRENDIZAJE

Unidad 1: Conceptos básicos

Competencia específica a desarrollar	**Actividades de Aprendizaje**
Dominar los conceptos relacionados con el análisis, diseño e implementación de aplicaciones para resolver problemas a través de la computadora	• Investigar la metodología para resolver los problemas a través de la computadora • Realizar un mapa conceptual sobre la metodología para resolver problemas a través de la computadora • Reconocer los conceptos básicos: algoritmo, programa, programación, paradigmas de programación utilizando mapas conceptuales, mentales, cuadros sinópticos, etc. • Conocer y dominar el uso de los operadores aritméticos y las funciones matemáticas

Unidad 2: Clases y objetos

Competencia específica a desarrollar	**Actividades de Aprendizaje**
Diseñar clases e implementar objetos cumpliendo las reglas de la programación orientada a objetos	• Programar clases con atributos públicos para exponer y comprender la vulnerabilidad de los datos. • Proteger los atributos con modificadores de acceso privados • Reunir dentro de una clase los miembros necesarios para resolver un problema en particular, y así implementar el encapsulamiento.

	• Instanciar objetos para identificar el nacimiento y muerte de los mismos. • Identificar operaciones que puedan ser realizadas por los objetos

Unidad 3: Introducción a la programación orientada a objetos en C#

Competencia específica a desarrollar	**Actividades de Aprendizaje**
Conocer las características principales del lenguaje de programación orientado a objetos. Codificar algoritmos en un lenguaje de programación orientado a objetos. Compilar y ejecutar programas.	• Conocer el entorno de un lenguaje de programación • Manejar la consola para compilar y ejecutar programas. • Realizar un mapa conceptual sobre los tipos de software y los conceptos básicos de programación. • Buscar y analizar información necesaria para instalar y configurar el compilador del lenguaje de programación a utilizar. • Mostrar al estudiante programas completos de menor a mayor grado de dificultad y con base en cada una de las instrucciones que los componen enseñar la sintaxis del lenguaje.

Unidad 4: Control de flujo

Competencia específica a desarrollar	**Actividades de Aprendizaje**
Construir programas utilizando estructuras condicionales y repetitivas para aumentar su funcionalidad.	• Realizar una investigación sobre el funcionamiento y aplicación de las estructuras de selección y de repetición. • Diseñar programas donde se utilicen las estructuras de repetición y selección. • Construir programas que implementen métodos o funciones.

Unidad 5: Métodos

Competencia específica a desarrollar	Actividades de Aprendizaje
Construir programas utilizando subrutinas como módulos para aumentar su funcionalidad.	• Realizar una investigación sobre los diferentes tipos de métodos (coincidencias y diferencias) • Diseñar programas donde se utilicen métodos que reciban parámetros. • Diseñar aplicaciones con funciones que devuelvan un valor.

Unidad 6: Arreglos

Competencia específica a desarrollar	Actividades de Aprendizaje
Construir programas que utilicen arreglos unidimensionales y multidimensionales para solucionar problemas.	• Diseñar algoritmos que utilicen arreglos unidimensionales y bidimensionales. • Desarrollar programas para implementar las operaciones básicas en arreglos.

11.- FUENTES DE INFORMACIÓN

1. Cairo Osvaldo. (2005). Metodología de la Programación. Algoritmos, diagramas de flujo y programas 3ª. edición. Editorial Alfaomega.
2. Ceballos, Francisco Javier. (2012). Microsoft C#. Curso de programación. 2ª. edición. Editorial Alfaomega.
3. Ceballos, Francisco Javier. (2013). Enciclopedia de Microsoft Visual C# 4ª. edición. Editorial Alfaomega.
4. Deitel y Deitel. (2007). Cómo Programar en C#. 5ª. edición. Prentice Hall.
5. López Román, Leobardo. (2013). Metodología de la Programación Orientada a Objetos 2ª. Edición. Editorial Alfaomega.
6. López Takeyas, Bruno. (2016). Curso de programación orientada a objetos en C# .NET. Ejemplos con aplicaciones visuales y de consola. Editorial Alfaomega.
7. López Takeyas, Bruno. (2016). Fundamentos de programación. 9 junio 2016, de Instituto Tecnológico de Nuevo Laredo. Sitio web: http://www.itnuevolaredo.edu.mx/takeyas/Materias/Fundamentos%20de%20Programacion/index.htm
8. López Takeyas, Bruno. (2014). Introducción a la ISC y al diseño de algoritmos 2ª. edición. Editorial Pearson.
9. López Takeyas, Bruno. (2016). Programación orientada a objetos en C#. 9 junio 2016, de Instituto Tecnológico de Nuevo Laredo. Sitio web: http://www.itnuevolaredo.edu.mx/Takeyas/Materias/POO/index.htm
10. Martín, Flowler & Kendall Scott. (2000). UML Gota a Gota. Addison Wesley.

11. Ramírez, Felipe. Introducción a la Programación, Algoritmos y su Implementación en Vb.Net C# Java y C++. 2a. edición, Alfa Omega.

12.- PRÁCTICAS PROPUESTAS

- Investigar la metodología para resolver problemas a través de la computadora.
- Analizar, diseñar e implementar aplicaciones orientadas a objetos
- Solucionar problemas con algoritmos a partir de enunciados proporcionados por el profesor.
- Crear, compilar y ejecutar programas de consola en C# mediante Microsoft Visual Studio.
- Las prácticas puedes accederse y descargarse de http://www.itnuevolaredo.edu.mx/takeyas/Apuntes/Fundamentos%20de%20Programacion/Practicas/index.htm