



UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS
FACULTAD DE INGENIERÍA

SYLLABUS
FACULTA DE INGENIERÍA

NOMBRE DEL DOCENTE:

ESPACIO ACADÉMICO: PROGRAMACION BÁSICA

Obligatorio (X) : Básico (X) Complementario ()
Electivo () : Intrínsecas () Extrínsecas ()

CÓDIGO: 2

NUMERO DE ESTUDIANTES:

GRUPO:

NÚMERO DE CREDITOS: 3

TIPO DE CURSO: TEÓRICO PRACTICO TEO-PRACTICO: ☒

Alternativas metodológicas:

Clase Magistral (x), Seminario (), Seminario – Taller (), Taller (x), Prácticas (x), Proyectos tutoriados(), Otro: _____

HORARIO:

DIA	HORAS	SALON

I. JUSTIFICACIÓN DEL ESPACIO ACADÉMICO

Competencias del perfil a las que contribuye la asignatura:	Esta asignatura contribuye al desarrollo de la competencia "Resuelve problemas computacionales algorítmicamente" que se encuentra en el dominio de "programación" del área "básicas de ingeniería" del proyecto curricular de ingeniería Electrónica.
Contribución a la formación:	En esta asignatura se establece las bases del pensamiento algorítmico formal que constituye uno de los pilares de la disciplina y contribuye a los dominios de desempeño profesional definidos en el perfil. A través de esta, se pretende mostrar al estudiante de manera práctica, la evolución de los lenguajes y paradigmas que han surgido alrededor de la programación, la adquisición de los conceptos básicos acerca de la estructura y funcionamiento del computador, así como el desarrollo del pensamiento algorítmico formal fortaleciendo sus habilidades en el desarrollo de programas computacionales. Estas habilidades se reconocen como claves dentro del dominio del perfil de "Programación".
Puntos de apoyo para otras asignaturas:	estructura lógica conceptual basada en paradigmas de programación.

	<ul style="list-style-type: none"> Herramienta fundamental para Programación orientada a objetos, Programación avanzada y Modelos de programación. Herramienta fundamental para ingeniería de software. Herramienta fundamental para bases de datos Herramienta fundamental para Redes Herramientas para Ciencias de la computación
Requisitos previos:	Lógica
II. PROGRAMACION DEL CONTENIDO	
OBJETIVO GENERAL	
<p>Presentar al estudiante, elementos fundamentales que le permitan tener claridad acerca de la evolución de la programación, de tal manera que pueda obtener soluciones a problemas sencillos apoyados en un computador, un lenguaje de programación y un paradigma en donde el alumno sea capaz de enfrentarse situaciones o problemas más complejos en las que debe identificar los elementos y estados involucrados generar modelos para su representación y manipulación algorítmica. Debe ser capaz de diseñar soluciones para los problemas, validar su corrección e implementar prototipos para ellas utilizando un lenguaje de programación de tipo estructurado.</p>	
OBJETIVOS ESPECÍFICOS	
<ol style="list-style-type: none"> Conocer la evolución de los lenguajes, los paradigmas y de la computación. Evidenciar de manera clara y concreta la evolución de la programación con relación a la evolución del computador. Identificar la estructura de un computador. Desarrollar el concepto de algoritmo y aplicarlo en la solución de programas sencillos Solucionar problemas elementales utilizando la lógica computacional Resolver problemas sobre el sistema computacional con la ayuda de un lenguaje de programación. Reconocer la sintaxis básica del lenguaje de programación escogido (preferiblemente C# o Java). 	
COMPETENCIAS DE FORMACIÓN:	
Competencias que compromete la asignatura:	El estudiante está en capacidad de pensar ordenadamente para modelar una solución a un problema haciendo uso de la algoritmia, expresando esta solución en un lenguaje computacional.
Competencias específicas de la asignatura:	Competencias Nucleares <ul style="list-style-type: none"> Utiliza adecuadamente el concepto y la abstracción de los sistemas numéricos en la solución de problemas computacionales. Localiza históricamente los diferentes momentos en la evolución de los sistemas computacionales. Identifica los diversos componentes de un sistema computacional. Representa soluciones de problemas aplicando el concepto de Algoritmo. Modela, implementa y evalúa problemas cuya solución algorítmica requiere el uso de las diferentes estructuras de control. Modela, implementa y evalúa problemas descomponiéndolos en subproblemas que permitan una solución más simple o la reutilización de soluciones. Resuelve problemas que requieren aplicar el concepto de recursividad. Define e implementa tipos de datos abstractos. Modela, implementa y evalúa mecanismos para el manejo dinámico de memoria y persistencia.
Competencias Transversales a las que contribuye la asignatura:	<ul style="list-style-type: none"> Es capaz de discernir que tecnología debe utilizar para la resolución de problemas particulares. Comunica ideas de forma clara oralmente o mediante la presentación de

	<p>documentos escritos.</p> <p>Actúa estratégicamente dentro de un grupo de trabajo para el desarrollo de proyectos.</p>
Programa sintético:	<ol style="list-style-type: none"> 1. Reconocer la estructura y funcionamiento del computador. <ol style="list-style-type: none"> 1.1. Sistemas numéricos: Sistema binario, hexagecimal y octal. 1.2. Conversiones entre sistemas. Números de precisión finita. 1.3. Representación de números negativos en base 2. 1.4. Representación de número punto flotante en base 2. 1.5. Operaciones. 1.6. Desarrollo histórico del "Hardware": El ábaco, Maquinas de Pascal, Leibniz, Babbage, Turing. 1.7. Primeros computadores: Mark1, ENAC, EDSAC, UNIVAC 1, Von Newman. 1.8. El computador hasta hoy: Generaciones. 1.9. Evolución de los lenguajes de programación. 1.10. Estructura del computador: Procesador, memoria principal, memoria secundaria, E/S, buses 2. Conceptualizar y abstraer problemas. Desarrollo de algoritmos. <ol style="list-style-type: none"> 2.1. Concepto de algoritmo 2.2. Los diagramas de flujo como herramienta de modelación de algoritmos. 2.3. Pseudocódigo: Una herramienta de palabras útil. 2.4. Modelar un problema de solución secuencial 2.5. Diseñar una solución algorítmica secuencial 2.6. Analizar una solución algorítmica secuencial 2.7. Modelar un problema cuya solución involucra condiciones 2.8. Diseñar una solución algorítmica que involucra condiciones 2.9. Analizar una solución algorítmica que involucra condiciones 2.10. Modelar problema cuya solución involucra iteraciones 2.11. Diseñar solución algorítmica que involucra iteraciones 2.12. Analizar una solución algorítmica que involucra iteraciones 2.13. Modelar problema complejo cuya solución amerita el uso de descomposición 2.14. Diseñar una solución algorítmica basada en descomposición 2.15. Analizar una solución algorítmica basada en descomposición 3. Diseñar soluciones algorítmicas para problemas computacionales (<i>Basado en el lenguaje de programación escogido. En este caso se hace referencia al lenguaje de programación C# o Java.</i>) <ol style="list-style-type: none"> 3.1. Estructura de un programa en C#, restricciones, comentarios 3.2. Tipos de datos, variables y constantes: Caracteres, Booleanos, Reales, Enteros. 3.3. Operadores <ol style="list-style-type: none"> 3.3.1. Aritméticos: asignación suma, resta, multiplicación, división, módulo, incremento, decremento, y todos asociados con una variable en una cantidad determinada. 3.3.2. Bitwise: And, Or, Or exclusivo, complemento, desplazamiento a izquierda y derecha, combinaciones con el operador de asignación. 3.3.3. Relacionales: menor que, mayor que, menor o igual que mayor o igual que, igual, diferente. 3.3.4. Booleanos: para la estructuración de expresiones: Not, And, Or. Jerarquías de los operadores. 3.3.5. Propositiones. And, or, xor, tablas de verdad. 3.4. Implementar prototipo de solución algorítmica secuencial 3.5. Conversión entre tipos de datos 3.6. Funciones de lectura y escritura. 3.7. Arreglos y matrices. Definición, inicialización. 3.8. Implementar prototipo de solución algorítmica que involucra condiciones: if, if else, switch. 3.9. Implementar prototipo de solución algorítmica que involucra

	<p>iteraciones: for, while, do while.</p> <p>Estructuras de salto: break, continue.</p> <p>Implementar prototipo de solución algorítmica basada en descomposición</p> <p>Funciones: Parámetros por valor, retorno de valores, variables locales, globales y estáticas.</p> <p>Librerías de funciones.</p> <p>Funciones recursivas.</p> <p>Referencias: Parámetros de funciones por apuntador y por</p>
--	--

referencia.

3.16. Manejo de archivos: persistencia de datos y flujo de datos.

RESULTADOS DE APRENDIZAJE	
<ul style="list-style-type: none"> · Entender el significado global de la Informática. · Comprender cómo se representa la información en el interior de un computador. · Conocer la estructura funcional de un computador. · Comprender el concepto de programación y enumerar sus principales características. · Conocer los fundamentos de los traductores: compiladores e intérpretes. · Manejar adecuadamente los sistemas operativos más comunes en la actualidad. · Manejar adecuadamente un lenguaje de programación · 	
Metodología Pedagógica y Didáctica:	
<ul style="list-style-type: none"> · Asistencia a clases expositivas y de discusión · Elaboración y lectura de paper (documentación). · Se debe procurar incentivar el trabajo de grupo más que el trabajo individual. (se recomienda trabajar en grupos de dos o tres estudiantes) · Implementación y prueba de prototipos (programas) en laboratorio de computación 	
<p>Trabajo Presencial Directo (TD): trabajo de aula con plenaria de todos los estudiantes.</p> <p>Trabajo Mediado_Cooperativo (TC): Trabajo de tutoría del docente a pequeños grupos o de forma individual a los estudiantes.</p> <p>Trabajo Autónomo (TA): Trabajo del estudiante sin presencia del docente, que se puede realizar en distintas instancias: en grupos de trabajo o en forma individual, en casa o en biblioteca, laboratorio, etc.)</p>	
IV. RECURSOS	
<p>Medios y Ayudas:</p> <ul style="list-style-type: none"> · Aula normal con pizarrón para sesiones de cátedra y para sesiones de discusión. · Disponibilidad para acceder a proyector multimedia. · Laboratorio de computación, con un computador por alumno, para las sesiones de laboratorio; cada computador debe contar con el intérprete para el lenguaje de programación que se va a utilizar para validar los prototipos. · Página web para publicar material didáctico, guías de ejercicios, soluciones, tareas, etc. · Acceso fuera de clases a laboratorios de computación que cuenten con el intérprete para el lenguaje de programación que se va a utilizar para validar los prototipos, y con acceso a la página web del módulo. · Acceso al material bibliográfico recomendado · Asignación de una persona que tenga las plenas competencias del curso (monitor) para asesorar a los estudiantes en dudas durante las sesiones del laboratorio de computación. 	
BIBLIOGRAFÍA	
TEXTOS GUÍA	
<ul style="list-style-type: none"> · Cairó, Oswaldo. Metodología de la Programación. Editorial Alfa Omega. · Harvey M. Deitel y Paul J. Deitel, C# Como Programar, segunda edición. 	
TEXTOS COMPLEMENTARIOS	

- Tanenbaum, Andrew. Structured Computer Organization. Prentice Hall.
- Levine, Guillermo. Computación y Programación Moderna. Addison Wesley.
- Rodriguez C., Llana L.F, Martinez, R., Palao P., Pareja, C. Ejercicios de Programación Creativos y Recreativos en C ++. Prentice Hall.
- Harvey M. Deitel y Paul J. Deitel, C# Como Programar, segunda edición.
- Alfredo Weitzenfeld, Ingenieria de Software orientada a Objetos con UM. Java e Internet.

REVISTAS							
		Horas		Horas		Total Horas	Créditos
		Horas		Horas		Estudiante/semestre	
		profesor/semana		Estudiante/semana			
Tipo de Curso		(TD + TC)		(TD + TC +TA)		X 16 semanas	
DIRECCIONES DE INTERNET							
http://msdn.microsoft.com/es-es/library/1637x362(VS.80).aspx						192	3
http://www.mygnet.net/manuales/java//guia_java.1691							

V. ORGANIZACIÓN / TIEMPOS

Espacios, Tiempos, Agrupamientos:

Se recomienda trabajar una unidad cada cuatro semanas, trabajar en pequeños grupos de estudiantes, utilizar Internet para comunicarse con los estudiantes para revisiones de avances y solución de preguntas (esto considerarlo entre las horas de trabajo cooperativo).

PROGRAMA SINTÉTICO	SEMANAS ACADÉMICAS															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Reconocer la estructura y funcionamiento del computador.	x	x	x	x												
Sistemas numéricos	x															
Conversiones entre sistemas. Números de precisión finita.	x															
Representación de números negativos en base 2.	x															
Representación de número punto flotante en base 2.	x															
Operaciones.	x															
Desarrollo histórico del "Hardware"	x															
Primeros computadores	x															
El computador hasta hoy: Generaciones.			x													
Evolución de los lenguajes de programación.				x												
Estructura del computador: Procesador, memoria principal, memoria secundaria, E/S, buses				x												
Conceptualizar y abstraer problemas. Desarrollo de algoritmos.					x	x	x	x	x							
Concepto de algoritmo					x											
Los diagramas de flujo como herramienta de modelación de algoritmos.					x											
Pseudocódigo: Una herramienta de palabras útil.					x											
Modelar problema de solución secuencial						x										
Diseñar una solución algorítmica secuencial						x										
Analizar una solución algorítmica secuencial						x										
Modelar problema cuya solución involucra condiciones							x									
Diseñar solución algorítmica que involucra condiciones							x									
Analizar una solución algorítmica que involucra condiciones							x									
Modelar problema cuya solución involucra iteraciones							x									
Diseñar solución algorítmica que involucra iteraciones								x								
Analizar una solución algorítmica que involucra iteraciones								x								
Modelar problema complejo cuya solución amerita el uso de descomposición									x							
Diseñar solución algorítmica basada en descomposición									x							

Analizar una solución algorítmica basada en descomposición									x								
Diseñar soluciones algorítmicas para problemas computacionales										x	x	x	x	x	x	x	x
Estructura de un programa en C#, restricciones, comentarios										x							
Tipos de datos, variables y constantes: Caracteres, Booleanos, Reales, Enteros.										x							
Operadores										x							
Implementar prototipo de solución algorítmica secuencial											x						
Conversión entre tipos de datos											x						
Funciones de lectura y escritura.											x						
Arreglos y matrices. Definición, inicialización.												x					
Implementar prototipo de solución algorítmica que involucra condiciones: if, if else, switch.												x					
Implementar prototipo de solución algorítmica que involucra iteraciones: for, while, do while.												x					
Estructuras de salto: break, continue.												x					
Implementar prototipo de solución algorítmica basada en descomposición													x				
Funciones: Parámetros por valor, retorno de valores, variables locales, globales y estáticas.													x				
Librerías de funciones.													x				
Funciones recursivas.														x			
Referencias: Parámetros de funciones por apuntador y por referencia.															x		
Manejo de archivos: persistencia de datos y flujo de datos.																x	

VI. EVALUACIÓN			
	TIPO DE EVALUACIÓN	FECHA	PORCENTAJE
PRIMERA NOTA	Prueba oral/escrita para el grupo que el docente elabora. Informes de conceptos basado en análisis (Paper's) Pruebas orales/escritas rápidas (Quizes)	Semana 4 ó 5	10 %
SEGUNDA NOTA	Prueba escrita conjunta (para todos los grupos de la asignatura), elaborada por los docentes que imparten la asignatura. Pruebas orales/escritas rápidas (Quizes)	Semana 10 - 11	20 %
TERCERA NOTA	Guías de ejercicios resueltas Informes de conceptos basado en análisis (Paper's) Pruebas orales/escritas rápidas (Quizes) Prueba escrita	Varias fechas	10 %
CUARTA NOTA	Informe de desempeño en laboratorio (Para 3 o 4)	Varias fechas	20 %
PROYECTO	Informe de desempeño y sustentación de un prototipo funcional que evalúe las competencias exigidas.	Semana 16	10 %
EXAMEN FINAL	Prueba escrita conjunta (para todos los grupos de la asignatura), elaborada por los docentes que imparten la asignatura.	Semana 17 - 18	30%
ASPECTOS A EVALUAR DEL CURSO			
<ul style="list-style-type: none"> · Claridad y entendimiento de los conceptos. · Que se haya identificado correctamente el problema y que el modelo lo represente adecuadamente. · Que la solución diseñada resuelva el problema. · Apego a la formalidad y estándares requeridos. · Que el análisis de corrección sea exhaustivo. · Que el prototipo corresponda al algoritmo diseñado y no presente errores de sintaxis. · La asistencia a las clases magistrales y a los laboratorios. · El esfuerzo y dedicación en la resolución de problemas. · Que la documentación permita reconocer la forma en que se ha abordado el problema y la estructura del programa implementado. · En las pruebas escritas se consideran en forma parcial los aspectos considerados en proyectos de programación bajo problemas que requieren un menor tiempo de desarrollo y en una modalidad que no requiere uso del computador, así como la comprensión conceptual. 			